Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Desarrollo de un sistema de inteligencia de negocio aplicado a la gestión de viajes

Estudante: Laura Rodríguez Silva
Dirección: Laura M. Castro Souto

A Coruña, xuño de 2020.

Nunca podrás hundir este barco. Xoel López

Agradecimientos

En primer lugar quiero agradecer a mi familia y amigos por apoyarme durante todos estos años, ya que soy consciente de que no ha sido un camino fácil. Me gustaría hacer una mención en especial a las siguientes personas:

A mis padres, por enseñarme el valor del trabajo y darme la oportunidad de hacerlo todo, anteponiendo mis ilusiones y metas a las suyas.

A los miembros de mi hogar: Pardavila, por el apoyo incondicional, por alegrarse de cada éxito como si fuese suyo y animarme en cada derrota.

A mi hermana, mi cuñado y mi sobrina por darme luz en los momentos más oscuros y apoyarme en mis decisiones.

A mis pichis, porque ellos han estado en el día a día de este camino hasta el momento actual, y sin ellos no sería posible.

Finalmente a mi tutora, Laura, por guiarme a lo largo del proyecto.

Resumen

En el momento actual utilizar la información sobre la producción de una empresa es un factor clave para el éxito del negocio. Gracias a las herramientas de *Business Intelligence* podemos convertir los datos generados internamente en conocimiento útil para la toma de decisiones estratégicas.

En este proyecto se ha desarrollado un sistema de *Business Intelligence* relacionado con viajes, para ello se ha recopilado información que se almacenó en un *data warehouse* haciendo uso de una herramienta ETL (*Extract*, *Transform*, *Load*) y posteriormente usando un servicio de análisis para la explotación de datos.

Abstract

Nowadays using the information about the production of a company is key for business success. Thanks to Bussiness Intelligence practices we can turn internally generated data into useful knowledge for strategic decision-making.

A Bussiness Intelligence system related to travel has been developed in this project. For this purpose, information has been collected, curated, and stored in a data warehouse using an ETL tool (*Extract*, *Transform*, *Load*). Afterwards, we have developed and used an analysis service for data exploitation.

Palabras clave:

Data Warehouse

Inteligencia de negocio

ETL

KPI

Métrica

SQL

Keywords:

Data Warehouse

Business Intelligence

ETL

KPI

Metric

SQL

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introdución

En la actualidad las empresas generan una gran cantidad de datos, los cuales se pueden extraer y transformar en información que resulte relevante a los usuarios de la empresa. Esta información se ha ido convirtiendo de cientos a millones de datos con lo que la forma de almacenarlos y procesarlos ha ido cambiando.

Para analizar estos datos contamos con las prácticas de *Business Intelligence* (Inteligencia de Negocio), las cuales nos permite identificar los factores que causan un buen o mal funcionamiento a la empresa, detectar situaciones que están fuera de lo normal, así como predecir comportamientos futuros basándonos en datos históricos.

1.1 Contextualización

En concreto en este caso nos centraremos en un mundo de reserva de viajes *online*. Este proyecto tiene como objetivo aplicar las ventajas que la Inteligencia de Negocio nos ofrece para optimizar el proceso de toma de decisiones, y así, mejorar el funcionamiento de la empresa tanto a corto como a largo plazo. Para conseguir esto vamos a utilizar un modelo basado en tecnologías OLAP, es decir, a partir de una gran cantidad de datos sin estructura procederemos a procesarlos en campos correctamente definidos, y proporcionaremos acceso inmediato a ellos para su consulta y posterior análisis.

La pieza clave para procesar estos datos son los procesos ETL (*Extract*, *Transform*, *Load*) , que se ocupan de (1) extraer los datos de un sistema fuente, que en el caso de este proyecto se tratará como único origen, PostgreSQL; (2) realizar la limpieza y organización de los datos que finalmente (3) desembocarán en un sistema destino, SQLServer, una base de datos relacional. A través de la aplicación de estos procedimientos a los datos para una aplicación de reserva de viajes, podremos estudiar cuáles son los destinos más solicitados, la cantidad de viajes cancelados, o las fechas en las que más se reserva, entre otros factores, y así intentar mejorar las estrategias de negocio.

Para visualizar el análisis de los datos utilizaremos PowerBI, una herramienta que nos permitirá crear un cuadro de mando, que consiste en la agrupación de métricas y gráficos que permiten evaluar la gestión de los viajes, detectar comportamientos anómalos y tomar decisiones en consecuencia.

1.2 Objetivos

Los objetivos que se desean alcanzar en este proyecto son los siguientes:

1. Aplicar las técnicas de la Inteligencia de Negocio en el sector de la reserva de viajes.
2. Proporcionar un cuadro de mando para el estudio de los datos una vez procesados.
3. Sanear automáticamente la información (tratamiento de nulos, eliminación de datos duplicados, uniformidad de tipado, etc.).
4. Definir KPIs (*Key Performance Indicators*), esenciales para conseguir una buena calidad de la información.
5. Maximizar la rentabilidad de la empresa.

Con el alcance de los objetivos se puede lograr un cuadro de mando que ayude a la empresa a mejorar su rendimiento, y, hacer útil aquella información que se ha almacenado a través de los años y no se estaba aprovechando. Así como, ampliar conocimiento de las herramientas que se van a utilizar, tanto a nivel de usuario como para el equipo.

Capítulo 2

Conceptos básicos

En este capítulo se introducen algunos conceptos básicos y términos con los que se desarrolla el proyecto

2.1 Business Intelligence

La Business Intelligence (Inteligencia de Negocio) [1] es un término tecnológico que engloba datos, informática y análisis dentro de operaciones de negocios. Incluye procesos y métodos para optimizar el rendimiento. Las herramientas de BI analizan conjuntos de datos y los presentan en informes, cuadros de mando, gráficos o mapas para proporcionar al usuario una información detallada sobre el estado del negocio.

Business intelligence[2] nos ofrece muchas ventajas porque podremos conocer mejor nuestro negocio y cambiar en función de las necesidades de los clientes, ver el estado actual de los costes en los diferentes niveles de agregación y mejorar el entendimiento con los clientes transformando la información de manera que aporte valor al negocio.

Los objetivos principales de la BI son:

- Mostrar información que afecte al rendimiento de la empresa
- Mejorar la eficiencia
- Aumentar la rentabilidad
- Crear estrategias orientadas al futuro
- Prevenir escenarios futuros basándonos en datos históricos

2.2 Sistemas OLAP

El término OLAP[3](*On-line Analitical Processing*) define una tecnología basada en el análisis multidimensional de los datos. Son bases de datos orientadas al procesamiento analítico, lo que implica el procesamiento de grandes volúmenes de datos procedentes de distintas fuentes.

Propiedades de las bases de datos OLAP

Aunque este sistema nos ofrece múltiples propiedades solo vamos a destacar las mas importantes:

- Normalmente los accesos a los datos son solo de lectura
- Asíncronas, no se actualizan siempre en tiempo real
- Suelen alimentarse con información de bases de datos relacionales mediante un proceso de ETL
- Crear estrategias orientadas al futuro

2.3 *Data Warehouse*

Un *Data Warehouse* [3] o Almacén de Datos es un sistema de almacenamiento diseñado para contener datos extraídos de fuentes externas. Son utilizados para el análisis y creación de informes una vez filtrados. Son sistemas pensados para almacenar grandes volúmenes de datos sobre información histórica de las empresas, de forma que las consultas sobre la base de datos sean lo mas eficientes posible.

Uno de los primeros autores en escribir sobre los almacenes de datos fue Bill Inmon.El científico inglés denominado por muchos el padre del *Data Warehouse* definió el sistema como:

- *Orientado:* Debe estar diseñado para mostrar un tema concreto.
- *Integrado:* Puede integrar datos de diferentes fuentes de datos.
- *No volátil:* Una vez almacenados los datos, se convierten en información de lectura, no se modifican ni se eliminan.
- *Variable en el tiempo:* Se deben poder almacenar datos históricos.

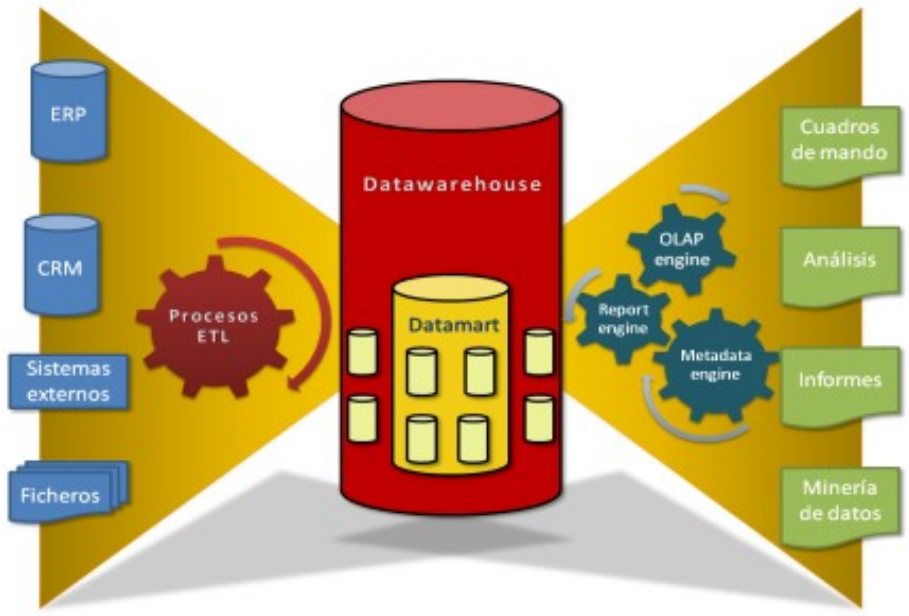


Figura 2.1: Data Warehouse

2.4 Procesos ETL

Los procesos ETL (*Extract*, *Transform*, *Load*) son un tipo de integración de datos que hace referencia a los tres pasos (extraer, transformar, cargar), que se utilizan para mezclar datos de múltiples fuentes y construir un almacén de datos.

Extracción Es la primera etapa de ETL, en ella se extraen los datos ubicados en distintas fuentes de las empresas. En este paso se convierten dichos datos a un único formato quedando así preparados para su posterior transformación. Una vez se encuentren recopilados se debe hacer un tratamiento de limpieza o depuración que será útil para evitar errores.

Transformación [4] Como su propio nombre indica los datos se transforman en la estructura que hayamos definido en nuestro *data warehouse*. Este paso incluye validar las reglas de negocio, normalización, homogeneización de códigos, cambio de formatos, así como la ordenación, filtrados y agregados.

Carga En este último paso se realiza la carga de datos ya transformados en el almacén de datos. Esta carga se puede realizar en el momento o de forma programada. Aprovechando de esta manera los momentos en los que la base de datos dé soporte a menos usuarios para ejecutar el proceso.

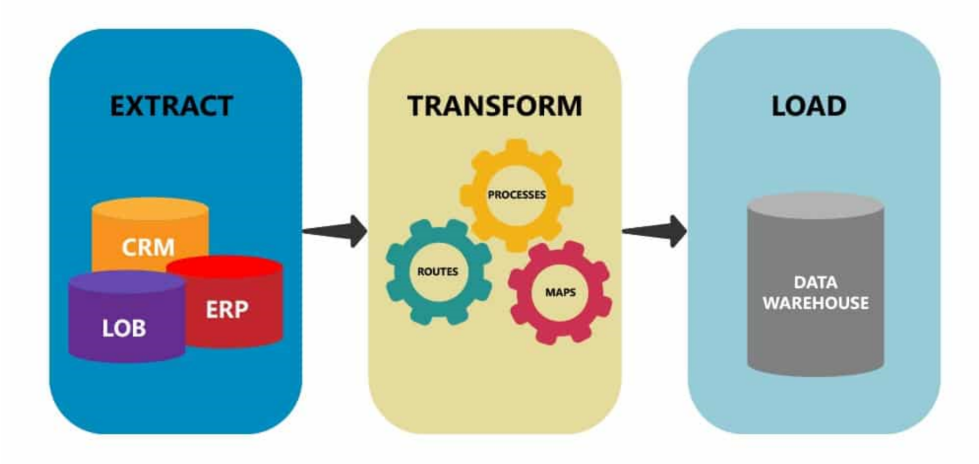


Figura 2.2: Proceso ETL

2.5 Modelado de datos

2.5.1 Modelo estrella

El modelo en estrella[5] , mostrado en la figura 2.3, es un tipo de diseño de base de datos orientado a un *data warehouse*. Su característica principal es la distinción entre dimensiones y hechos. Las **tablas de hechos** son los datos utilizados para análisis. Representan las medidas que se quieren analizar para tomar decisiones.Por el contrario, **las dimensiones** representan aspectos de interés, por los que se podrán filtrar. Su principal función es proveer un contexto lógico de negocio que es algo que las tablas de hechos no hacen.

2.5.2 Modelo copo de nieve

El modelo copo de nieve 2.4 es una variación del modelo estrella, con la diferencia de que este modelo consta de una tabla de hechos que está conectada a diversas tablas de dimensiones, que pueden estar a su vez conectadas con otras tablas de dimensiones.

2.5.3 Ventajas y desventajas de los modelos

En muchos de los factores nos favorece un esquema estrella ya que es simple y veraz para ser usado en análisis multidimensionales. Es simple desde el punto de vista del usuario final ya que las consultas no son complicadas. Sin embargo, el modelo copo de nieve tiene la ventaja de que las tablas de las dimensiones están normalizadas, de esta manera se evita redundancia y con ello ahorramos espacio.

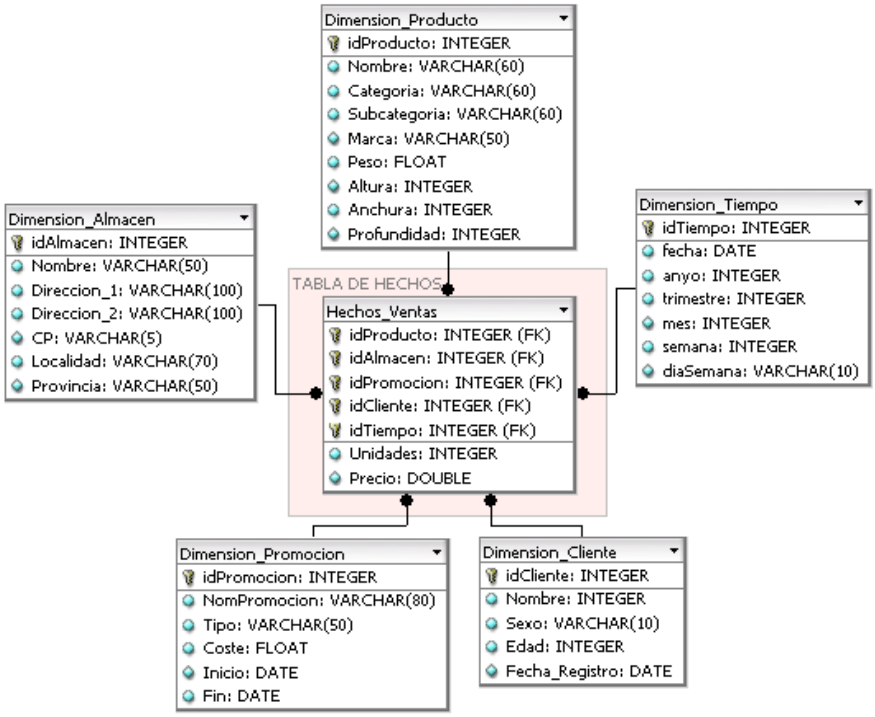


Figura 2.3: Ejemplo de modelo estrella

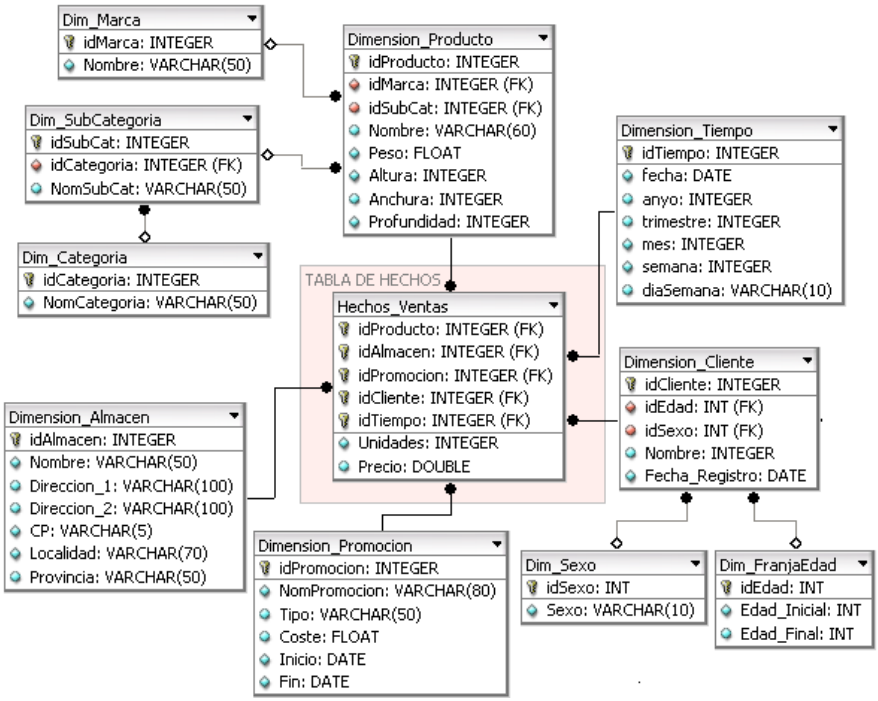


Figura 2.4: Ejemplo de modelo copo de nieve

2.6 *Key Performance Indicators*

Los KPI [6]*(Key Performance Indicator)* son métricas para identificar el rendimiento de una acción o estrategia. Con ellos podemos reflejar el estado actual de un negocio con respecto a un área en concreto, y a partir de ahí, actuar para optimizar y mejorar el rendimiento de la empresa. Para definir un KPI debemos establecer unos criterios básicos que nos permitan medir la evolución de una empresa. Para ello debe ser específico, cumplir un objetivo concreto que no dé lugar a confusión.

Tiene una gran cantidad de ventajas que nos ayudan a hacer mas comprensible lo que queremos mostrar en nuestro cuadro de mando:

- Medición constante, incluso en tiempo real, lo que nos permite actuar de forma rápida.
- Adaptación del negocio a los cambios continuos de mercados, clientes u oportunidades.
- Tranquilidad de directivos, inversores o altos cargos relacionados con el negocio.

2.7 Arquitectura de los sistemas de explotación

En la imagen 2.5 se muestran los principales elementos que interaccionan con los sistemas de explotación. Se divide en 3 partes: sistemas operacionales, sistemas de explotación con el *backend* y el *frontend* y finalmente usuarios u otros productos.

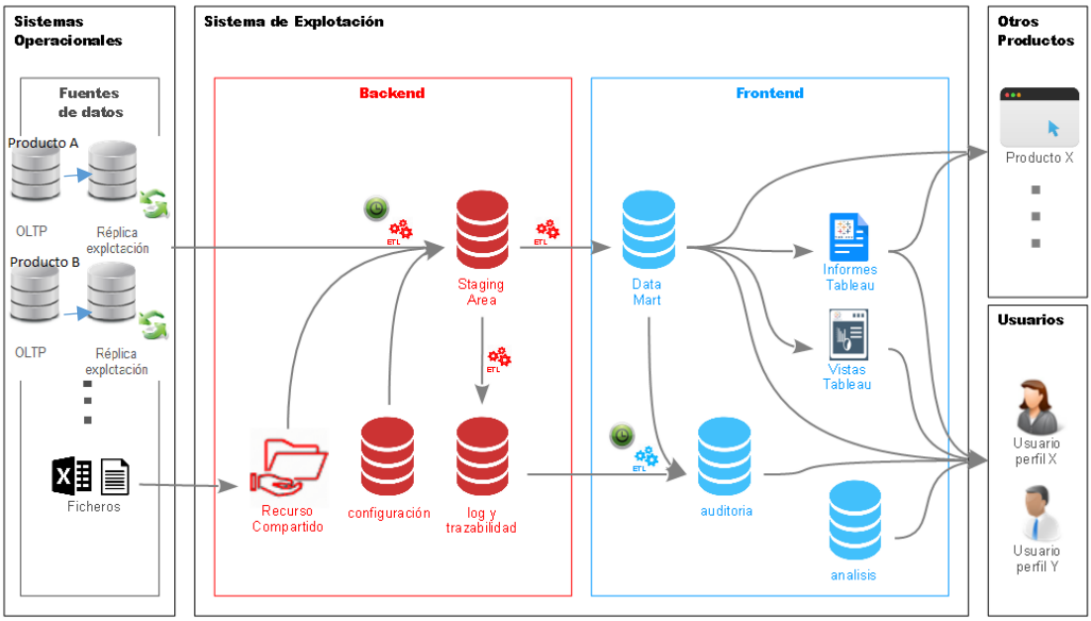


Figura 2.5: Sistema de explotación

2.7.1 Sistemas operacionales

Los sistemas operacionales están compuestos por ficheros y la base de datos OLTP *(Online Transaction Processing)*; es un tipo de procedimiento que facilita el procesamiento y la entrada de datos destinada a la administración de las operaciones transaccionales.

Estas bases de datos son un repositorio de datos de las aplicaciones corporativas que nos servirán como fuente de datos.

2.7.2 *Backend*

El *backend* está conformado por varias bases de datos:

Logs y trazabilidad En ellos se almacena toda la información acerca de los *logs* de ejecución, errores, *warnings* y rendimiento de los procesos ETL.

Configuración Recoge los datos de configuración global de los diferentes procesos ETL del sistema de explotación.

Staging Area Se utiliza para el proceso de transformación. Aquí realizamos la transformación de los datos haciendo tareas de limpieza, integración y unificación de datos que se cargarán en la base de datos.

2.7.3 *Frontend*

En el *frontend* también contiene varias bases de datos:

Data Mart Da servicios a usuarios finales.

Auditoría Este repositorio incluye información acerca de las incidencias de datos que tratamos en los diferentes procesos de ETL.

Análisis Es una agregación de la información del *Data Mart*, previa a los cálculos que se realizan sobre los datos. Esta base de datos se utilizará cuando realicemos varias consultas con bajo rendimiento.

Capítulo 3
Tecnologías

En este capítulo trataremos algunas de las tecnologías dentro del mundo de *Business Intelligence*. Haciendo una breve referencia a las que están bien posicionadas y las que emplearemos para este proyecto

3.1 Diagrama Gartner 2020

Para una visión global de las soluciones dentro del mercado de *Business Intelligence* veremos el *Magic Quadrant for Analytics and Business Intelligence Platforms*; [7] (figura 3.1) realizado por la empresa Gartner tras una investigación de mercado de la industria tecnológica. El cuadrante nos permite saber que posición ocupa un proovedor y en que punto de desarrollo esta respecto al mercado.

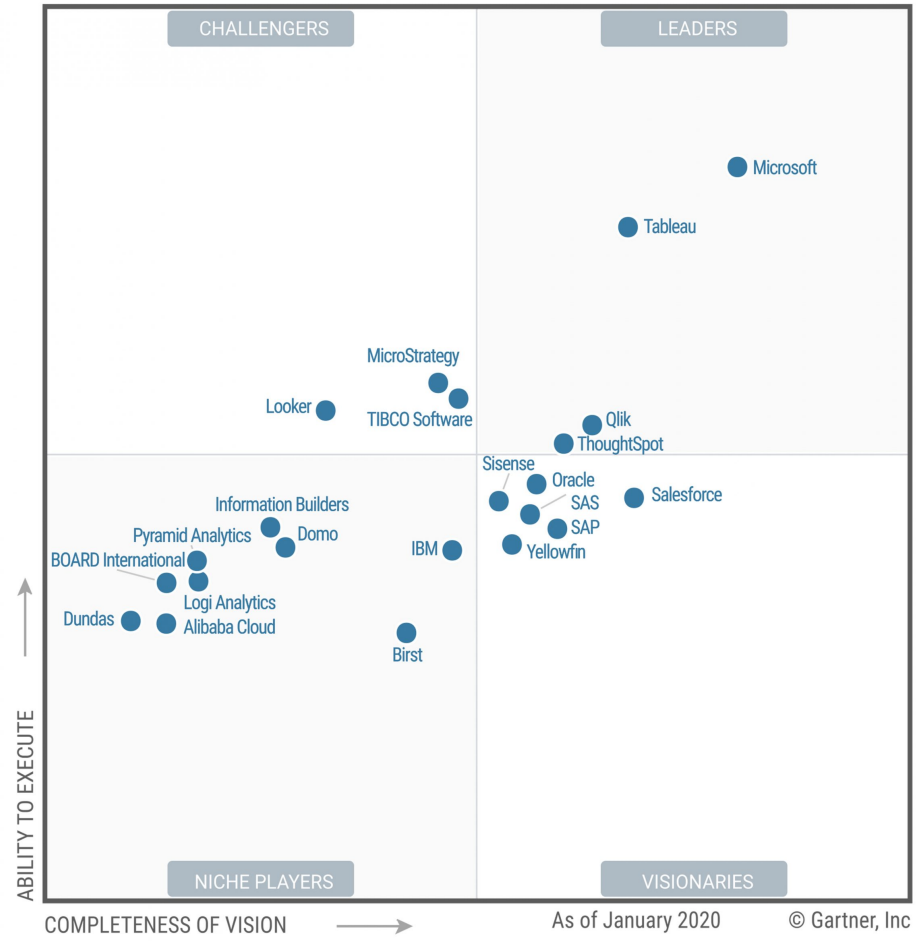
A partir de esta visión general nos centraremos en aquellos productos con funcionalidades específicas para el diseño:

Qlik Esta herramienta [8] nos permite tener varias fuentes de datos, acceso inmediato en tiempo real y un alto factor de compresión entre otras cosas, pero para trabajar con Qlik es recomendable tener un servidor dedicado porque consume muchos recursos. Actualmente esta herramienta se está quedando obsoleta respecto a otras que están en pleno auge y su precio es bastante elevado.

Tableau [9] es intuitiva y fácil de utilizar con todo tipo de gráficas e indicadores. Nos ofrece la posibilidad de visualización en todo tipo de dispositivos, y un despliegue en la nube de Tableau o de terceros como Amazon o Azure, aprovechando las ventajas que proporcionan esas plataformas. Actualmente está liderando el mercado junto con Microsoft, a pesar de que es una herramienta con un coste muy elevado.

Power BI permite recoger datos de diversas fuentes, las locales o aquellas que residen en la nube, como es el caso de Azure en Microsoft. Es más fácil de implementar por un usuario

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms



Source: Gartner (February 2020)

Figura 3.1: Magic Quadrant for Analytics and Business Intelligence Platforms

de negocio debido a paquetes de Office a los que los usuarios están acostumbrados. Es la herramienta que vamos a utilizar debido a sus grandes funcionalidades y a su coste reducido.

3.2 PostgreSQL

PostgreSQL es una base de datos relacional *open-source*. Las características que ofrece: estabilidad, potencia o fácil administración entre otras la hace una base de datos potente. Nos ofrece un sistema con concurrencia multiversión, es decir, se puede acceder a las mismas tablas sin necesidad de bloqueos.

Ventajas

PostgreSQL no es solamente una base de datos fácil de administrar, sino que tiene distintas ventajas:

- Gratuito: Es un gestor de bases de código libre con lo cual podemos instalarlo y utilizarlo las veces que queramos y sin límite de dispositivos
- Multiplataforma: Es compatible con muchas tecnologías y sistemas operativos.
- Escalabilidad: Es una herramienta que se configura de forma individual según los recursos de los que se dispone, se puede ajustar la CPU y la memoria para hacerla lo mas óptima posible adaptada a nuestras necesidades.
- Gráfica: Incorpora una herramienta gráfica que nos permite administrar una base de datos de manera intuitiva y fácil.
- Robustez y fiabilidad: Cumple el protocolo ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Con esto se garantiza la fiabilidad del sistema.

Desventajas

No tiene soporte oficial, pero si cuenta con foros oficiales, donde los usuarios exponen sus dudas y responden a otros usuarios.

3.3 SQLServer

Microsoft SQL Server es una alternativa de Microsoft frente a otros gestores de bases de datos. Este sistema gestor está desarrollado como un servidor que presta servicio a otras aplicaciones. Su principal característica es la alta disponibilidad, permite mucho tiempo de

actividad con una rápida conmutación sin que se vean afectados los recursos de memoria del sistema. No obstante, también consta de un entorno gráfico para la administración que nos permite el uso de DDL (*Data definition language*) y DML (*Data manipulation language*) gráficamente.

Este gestor nos ofrece trabajar en modo cliente-servidor, y nos permite separar los datos que se alojan en los servidores frente a los clientes que solo acceden a la información. Sin embargo, no nos podemos olvidar que utiliza mucha memoria RAM y que esta diseñado para utilizarse en servidores basados en Windows.

3.4 Pentaho

Pentaho [10] es una herramienta de BI que nos permite extraer, analizar y cargar datos con técnicas ETL. En este caso estamos ante un software *open source* que cuenta con herramientas de Big Data e IoT (Internet of Things). Actualmente muchas empresas la emplean para mejorar la capacidad de análisis y la toma de decisiones

Ventajas y desventajas

Una de las principales ventajas de esta herramienta es que es accesible sin coste y con grandes capacidades.Es importante destacar que es multiplataforma y esto nos permite usarlo en cualquier sistema operativo, o añadir algunas funcionalidades o módulos adaptándonos así a las necesidades de cada proyecto.

En cuanto a las desventajas es importante tener en cuenta que es una herramienta que carece de soporte oficial, con lo que la documentación en ocasiones puede ser insuficiente y tenemos que recurrir a otros usuarios en foros.[10] Otro de los fallos que se presenta es que es poco intuitiva, ya que fue creada por varios profesionales y existen varias visualizaciones.

Componentes

Pentaho Business Analytics Platform: Es una plataforma web que aloja informes o contenido propio de la web.

Pentaho Data Integration(PDI): La herramienta ETL de pentaho que nos permites extraer datos de distintas fuentes, transformarlos de la manera que nos sea útil y cargarlos en diferentes destino.

Pentaho Report Designer: Herramienta para generar informes configurando una fuente de datos.

3.5 Power BI

Power BI [11] es un servicio gratuito de análisis de negocio y visualización de datos. Con ella controlamos el rendimiento del negocio mediante *dashboards* o informes. Esta herramienta nos ofrece tanto un servicio de nube como una versión de escritorio descargable que ofrecen diferentes capacidades. En la actualidad cuenta con 5 millones de usuarios y es utilizado por una gran cantidad de empresas.

Power BI es una herramienta que ofrece una gran cantidad de información y es muy intuitiva si estás familiarizado con entornos de bases de datos.

Una parte a tener en cuenta a la hora de guardar nuestro cuadro de mando es como hacerlo, nos ofrece dos maneras: La primera, *.pbit*, no incluye ningún dato del sistema origen, es decir, se guarda solamente la estructura del informe. Por otro lado, *.pbix*, no solamente guarda la estructura del informe, sino que también almacena la información, en memoria, de las distintas fuentes de datos.

Beneficios

- Las funciones integradas de aprendizaje automático pueden ser valiosas para hacer predicciones.
- Visualización mediante plantillas para un entendimiento de los datos rápido y fácil.
- Cuenta con una interfaz intuitiva.
- Garantiza la seguridad de los datos ofreciendo controles de acceso.
- Permite la actualización de datos de manera programada.
- Permite crear diseños para dispositivos móviles.

Componentes

Power BI se compone de varias aplicaciones:

Power Query: Herramienta de conexión de datos que permite transformar, combinar y mejorar datos de varias fuentes.

Power Pivot: Herramienta de modelado de datos.

Power View: Visualización de datos que genera gráficos, mapas y elementos visuales.

Power Map: Para creación de imágenes 3D envolventes.

Power Q&A: Motor de preguntas y respuestas que permite hacer preguntas sobre datos en un lenguaje sencillo.

En base a la información presentada en este capítulo, se decidió usar SQLServer en lugar de PostgreSQl para el almacenamiento de los datos, ya que era una herramienta utilizada recientemente por la desarrolladora, motivo por el cual también se eligió utilizar Pentaho.

Power BI es una herramienta de licencia libre la cual está siendo demandada por las empresas dedicadas a este sector, por este motivo la alumna consideró interesante su uso para mejorar sus capacidades de cara al mundo laboral.

Capítulo 4

Metodología

La metodología nos sirve como una guía para estructurar y planificar un proyecto. Con ella se puede mejorar la productividad y la calidad de los productos finales, de igual modo, fomentar el trabajo en equipo.

4.1 Scrum

Scrum [12] es un proceso para gestionar la complejidad en el desarrollo de un producto y satisfacer las necesidades de los clientes. Es un marco simple de trabajo que promueve la colaboración en equipo para lograr desarrollos complejos.

Está basado en la auto-organización de equipos para lidiar con lo imprevisible y resolver problemas complejos adaptándose continuamente.

4.1.1 Roles

En términos generales tenemos 3 roles que nos podemos encontrar en un proyecto con varios usuarios:

Product Owner El interesado en el producto. La persona encargada de lograr el mayor valor del producto para los clientes, usuarios e implicados.

Scrum Master Responsable del funcionamiento de Scrum en la organización.

Development Team Encargados de desarrollar el sistema.

Sin embargo en nuestro caso se trata de un proyecto de Fin de Grado, el equipo esta exclusivamente formado por una persona que desempeña las responsabilidades de Scrum Master y Development Team

4.1.2 Sprints

Sprint es cada uno de los ciclos o iteraciones que vamos a tener dentro de un proyecto *Scrum*. Nos va a permitir tener un ritmo de trabajo con tiempo, siendo la duración máxima de un *sprint* 4 semanas.

En cada *sprint* lo que vamos a conseguir es lo que denominaremos un entregable ,o incremento del producto que aporta valor al cliente.

Cuando se trata de un proyecto como el que vamos a realizar, el primer *sprint* comienza con la reunión de planificación, en ella decidimos qué vamos a hacer y cómo lo vamos a hacer.

4.1.3 Eventos

Los eventos de Scrum se utilizan para minimizar la necesidad de reuniones no definidas y fomentar en el equipo la comunicación y colaboración. Todos los eventos tienen una *TimeBox*: una vez que se inicia un evento tiene duración fija que no se puede modificar. Los siguientes eventos pueden terminar si se logra el propósito del evento dentro del tiempo estimado.

Estas reuniones son:

Sprint planning En esta reunión se planifica el siguiente *sprint*, tratando la duración de las actividades y los recursos necesarios para su ejecución. El *Product Owner* y el *Scrum Master* son los que dividen el proyecto en etapas y tareas asignando el trabajo a los distintos miembros del equipo.

Daily scrum Con esta reunión comprobaremos que no existan obstáculos que impidan la ejecución de las actividades en progreso. Esta reunión no suele durar más de 15 minutos a primera hora de la jornada.

Sprint review Reunión de cierre para exponer el resultado final a todo el equipo. En este momento se hablará de los cambios a realizar.

Sprint retrospective Esta metodología recomienda la celebración de reuniones privadas entre el *Scrum Master* y el equipo, con el fin de revisar cómo se ha gestionado el proyecto finalizado.

4.1.4 Artefactos Scrum

Los artefactos son los recursos que cimientan la productividad y la calidad de cualquier proyecto:

Historias de usuario: Definición de una funcionalidad que debe cumplir el producto.

Product Backlog: Es una lista que define todos los requisitos del proyecto por parte del

Product Owner. Por definición esta lista nunca esta completa, va cambiando constantemente con el entorno y el producto. Para ello tenemos los siguientes atributos: descripción, ordenación, estimación y valor,

Sprint BackLog: Son los elementos seleccionados para ser ejecutados durante el *sprint* en curso. Es decir, el *Sprint Backlog* es un plan para entregar lo programado al final del *sprint*

Incremento: Es el resultado de cada *sprint*. Un incremento tiene que estar terminado, es decir, listo para ser usado.

En el caso de nuestro proyecto se realiza utilizando la herramienta *Trello* que explicaremos en el siguiente apartado 4.1.5

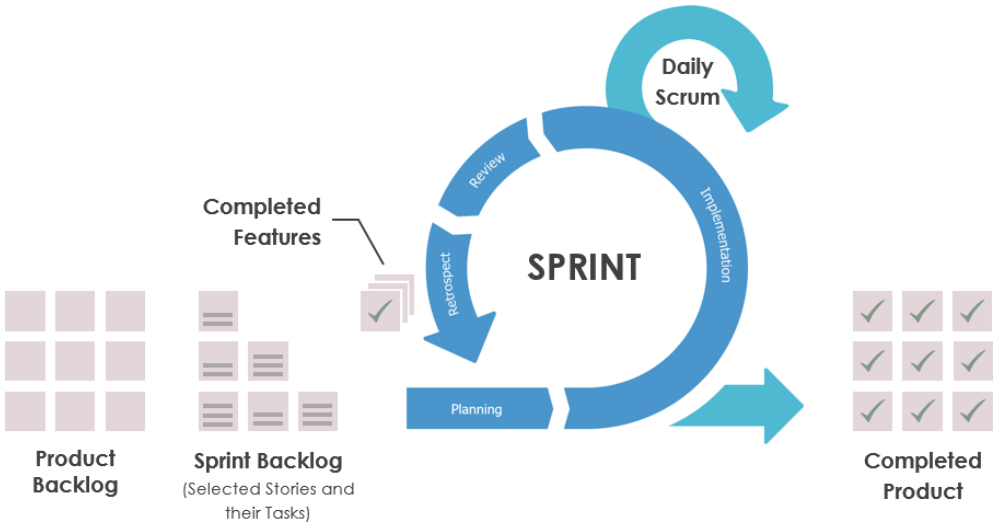


Figura 4.1: Scrum

4.1.5 Trello

Trello [13] una aplicación de gestión de proyectos muy utilizada a nivel tanto profesional como personal. Es una aplicación que sirve para gestionar tareas permitiendo organizar el trabajo en grupo de forma colaborativa, mediante tableros virtuales compuestos de listas de grandes tareas en formas de columnas.

Es perfecta para la gestión de proyectos ya que puede representar distintos estados y compartirlos con personas que formen parte del proyecto.

Para cada sprint se fueron creando tableros con sus correspondientes tareas cumpliendo las fechas cada 15 días.

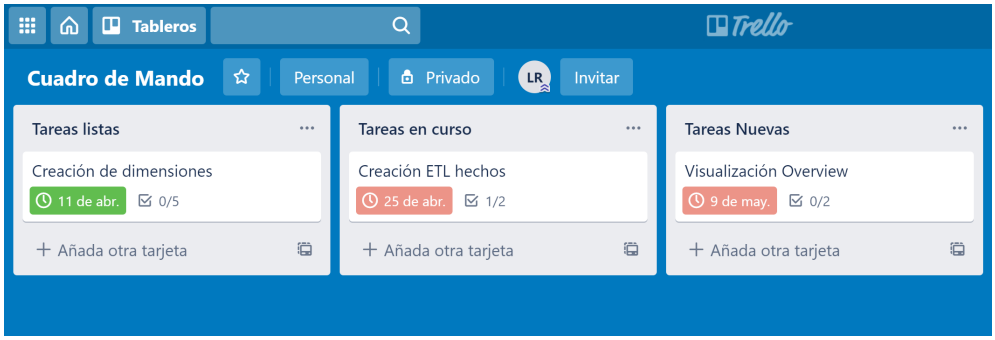


Figura 4.2: Ejemplo en Trello

En la imagen 4.2 mostramos un pequeño ejemplo de cómo gestionamos algunos de los *sprint*. Los que ya se han realizado, los que están actualmente en desarrollo y los que empezaremos próximamente.

En la imagen 4.3 podemos observar las tareas dentro de un *sprint*.

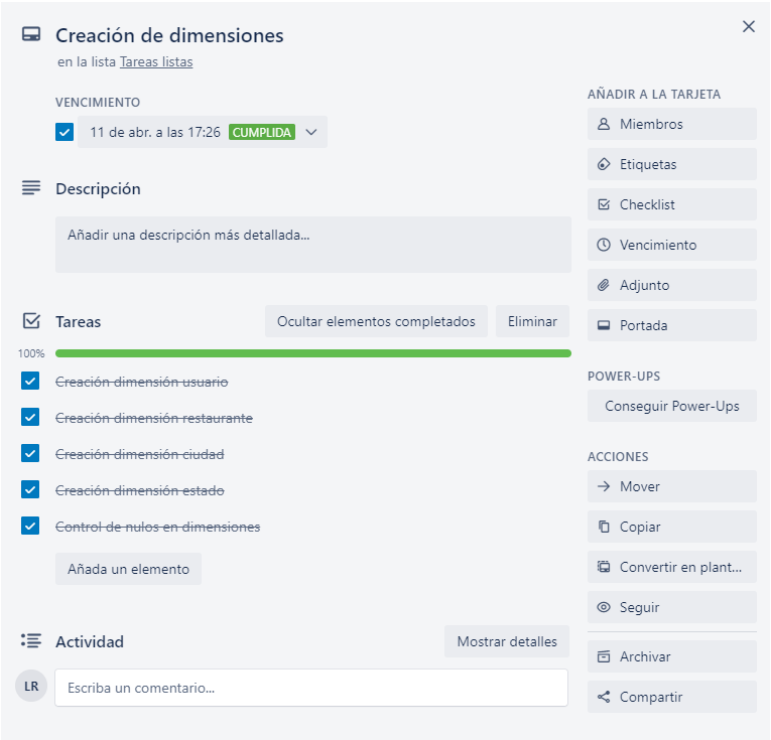


Figura 4.3: Tareas en Trello

Al tratarse de un proyecto en el que los requisitos estaban establecidos desde un principio, y en el que no iban a existir reuniones con el cliente en las que se cambiaran dichos requisitos, se ha adaptado la aplicación de la metodología scrum a estas necesidades. Las historias de usuario no han sido necesarias, ya que se ha usado una planificación fija en base a las tareas

marcadas desde un principio. En la metodología scrum, esta planificación se hace en base a funcionalidades que conllevarán la realización de tareas para su alcance. A pesar de no seguir la metodología en todos sus ámbitos, ha servido para llevar un control sobre las tareas realizadas y el avance conseguido a lo largo del tiempo.

Capítulo 5

Planificación

En este capítulo se explica como se llevo a cabo la planificación del proyecto, describiendo los objetivos de cada *sprint*.

5.1 Distribución de tareas

5.1.1 Sprint 0

Aunque en el proyecto actual no existe un entregable en este *sprint*, se ha decidido mantenerlo para determinar el alcance del proyecto,ya que se trata de un trabajo de fin de grado en el que se ha establecido la base sobre la que realizarlo, sin un cliente que marque los objetivos.

Por lo que en esta iteración se ha realizado la definición de requisitos y formación tecnológica. Este *sprint* se considera como fase previa para la preparación del proyecto. Tiene como objetivo principal una reunión en la que se estudia el alcance del proyecto, los requisitos mínimos, las herramientas necesarias y la duración del mismo.

Posterior a la reunión se empiezan a estudiar las tecnologías.

5.1.2 Sprint 1: Análisis de requisitos y modelado de datos

Este *sprint* tratará de analizar la fuente de datos y modelar la información.

- Estudio de los datos origen de PostgreSQL.
- Creación del modelo copo de nieve.

5.1.3 Sprint 2: Modelo de datos

Se crean las distintas tablas en SQLServer siguiendo el modelo copo de nieve creado en el Sprint 1. Se crean las tablas de dimensiones y de hechos para su posterior tratamiento.

5.1.4 Sprint 3: Desarrollo del Data Warehouse - Dimensiones

Se crea la primera parte del *Data Warehouse*, las tablas de dimensiones partiendo del origen PostgreSQL y se cargan en SQLServer tras las transformaciones necesarias.

5.1.5 Sprint 4: Desarrollo del Data Warehouse - Hechos

En este *sprint* se continua construyendo el *Data Warehouse* centrándonos en la tabla de hechos, se realizan las transformaciones necesarias para evitar carga de datos inconsistentes en SQLServer. Una vez finalizado, se realizan pruebas manuales para asegurarnos que los datos se han cargado correctamente.

5.1.6 Sprint 5: Visualización - Overview

En el *sprint* 5 se comenzará cargando los datos desde nuestras tablas SQLServer a Power BI. Una vez completo realizamos la pestaña Overview con sus métricas correspondientes.

5.1.7 Sprint 6: Visualización- Booking Lead y Booking Request

En este sprint se realizan dos nuevas pestañas basadas en las reservas, por un lado *booking lead* y por otro *booking request* (explicadas en desarrollo) con sus correspondientes gráficos.

5.1.8 Sprint 7: Memoria

Para finalizar el desarrollo del proyecto se procede a realizar la presente memoria en la hemos explicado el trabajo realizado.

5.2 Seguimiento

En la figura 5.1 se muestra la línea base, es una fotografía inicial del proyecto, previa a empezar el seguimiento.

En la figura 5.2 se muestra el seguimiento, es decir, si las tareas han sufrido retraso en completarse respecto a la linea base. Podemos observar que en el sprint 3 y 4 se sufrió un retraso de una semana en cada uno, debido al desconocimiento sobre el concepto de carga incremental, lo que implicó una dedicación de tiempo al análisis provocando dicho retraso. Este desajuste en la planificación causó cambios en el tiempo estimado, sin embargo no afectó al alcance, siendo la única consecuencia de esto un retraso totalmente asumible.

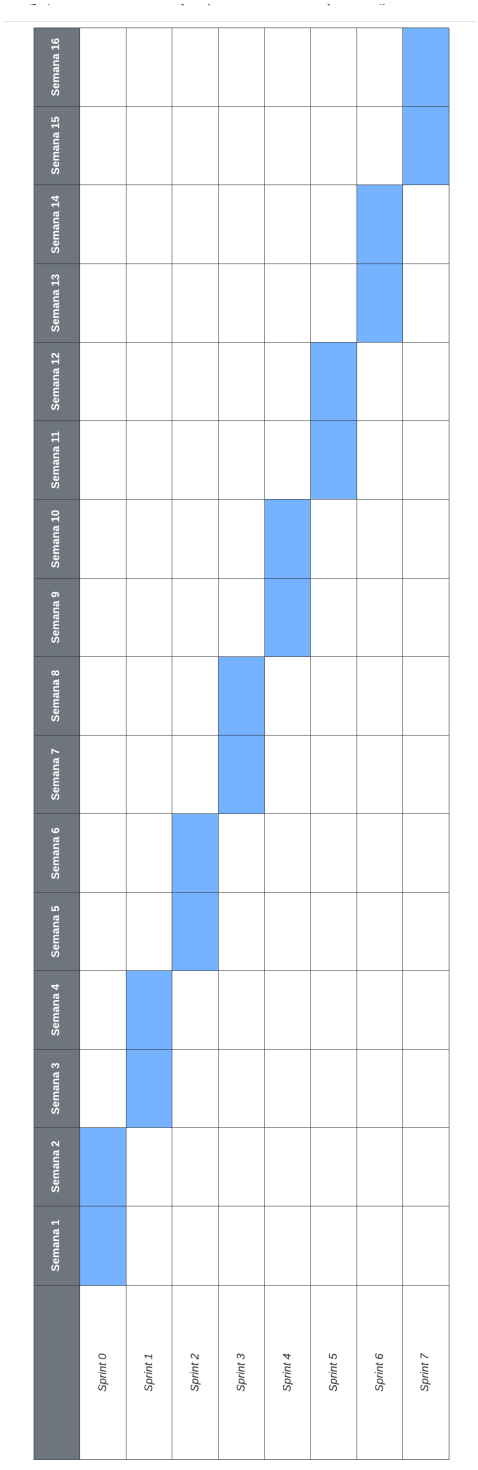


Figura 5.1: Diagrama de Gantt estimado

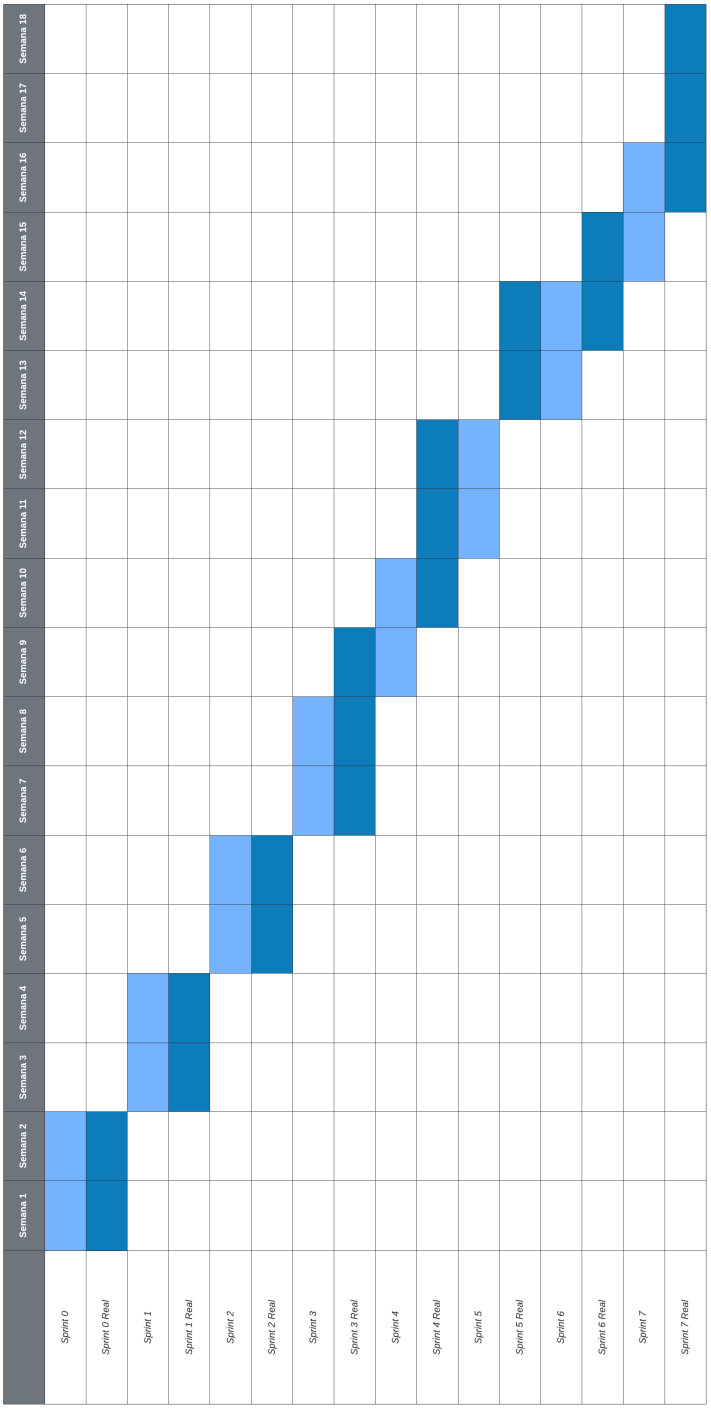


Figura 5.2: Diagrama de Gantt real

5.3 Costes

Teniendo en cuenta que se trata de un proyecto de fin de grado utilizamos el ordenador personal, con lo cual no necesitamos equipamiento extra. En cuanto a licencias todas las que utilizamos son gratuitas y por lo que no se añade un coste especial al proyecto. En relación a las horas de trabajo realizadas por el empleado, a la empresa le supondría un coste de 30 euros por hora incluyendo gastos, lo cual se traduce en un total de 10.800 euros explicados detalladamente en la tabla 5.1.

Sprint	*Propuesta*	*Fecha Inicial*	*Fecha Final*	*Horas*
0	Definición de requisitos y formación tecnológica	10/02/2020	23/02/2020	40
1	Análisis de requisitos y modelado de datos	24/02/2020	08/03/2020	40
2	Creación del modelo de datos	09/03/2020	22/03/2020	40
3	Desarrollo del Data Warehouse - Dimensiones	23/03/2020	12/04/2020	60
4	Desarrollo del Data Warehouse - Hechos	13/04/2020	03/05/2020	60
5	Visualización - Overview	04/05/2020	10/05/2020	20
6	Visualización - Booking Lead y Booking Request	11/05/2020	24/05/2020	40
7	Memoria	25/05/2020	15/06/2020	60
Esfuerzo total				360 h
Valoración en coste del esfuerzo en horas				10.800 €

Cuadro 5.1: Costes del proyecto

Capítulo 6

Desarrollo

En este capítulo se va a abordar el diseño e implementación del desarrollo desde el punto de vista de Pentaho, Power BI y SQLServer.

Partimos de los datos generados por una aplicación de reservas de viaje en la que un cliente puede reservar un itinerario, o publicar uno que haya realizado. El resto de clientes podrán acceder a dichos itinerarios y reservarlos si así lo desean. En caso contrario los agentes de *Booking* ofrecerán alternativas que se adapten en todo momento a sus necesidades.

Desde el punto de vista de la empresa esta reserva puede estar en distintos puntos:

Booking Lead: Es el primer paso de la reserva,el cual puede llegar a convertirse en *booking request.*

Booking Request: Al usuario se le ofrece un itinerario, puede decidir aceptarlo con lo cual la reserva se cerraría o puede cancelarlo.

Booking Closed: El usuario ha aceptado la reserva y se cierra

Booking Cancel: Cuando una reserva se cancela pasa a este estado. Puede ser que se haya cancelado, porque el usuario no esta conforme con las alternativas proporcionadas por los agentes, o bien por algún motivo personal del usuario (indisposiciones a poder realizar el viaje)

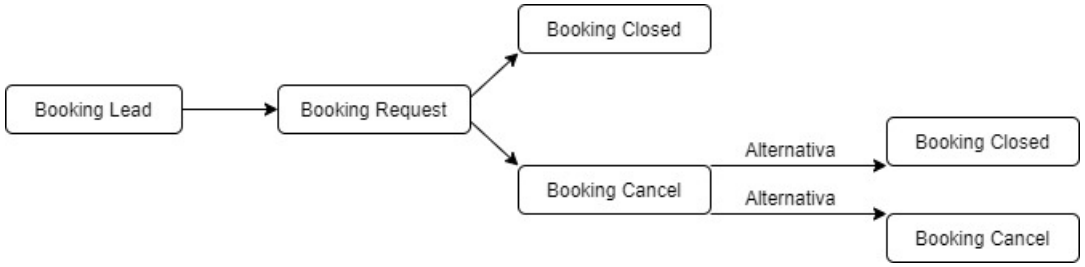


Figura 6.1: Estructura de las reservas

A continuación vamos a detallar cada *sprint* explicando los pasos que hemos seguido desde el origen hasta la visualización

6.1 Sprint 0: Definición de requisitos y formación tecnológica

Como se ha comentado anteriormente, en este primer *sprint* se realiza una reunión con el cliente para analizar los requisitos y el alcance del proyecto.

Tras la reunión, se lleva a cabo la formación de las tecnologías seleccionadas, este proceso se divide en dos fases. La primera de ellas centrada en las tecnologías de *backend*, en la que se estudia SQLServer, una herramienta con la que el equipo ya ha trabajado en otras ocasiones ,y la segunda Pentaho ,que se había tratado en otros proyectos, pero en la que no se tiene mucha experiencia. La fase en la que se tuvo que invertir más tiempo fue en estudiar las tecnologías utilizadas en el *frontend*, Power BI, ya que no se había trabajado nunca con ellas, eran completamente desconocidas.

6.2 Sprint 1: Análisis de requisitos y modelado de datos

6.2.1 Modelo copo de nieve

En esta fase se realizó el modelo analizando los datos que teníamos. En la imagen 6.2 se muestra el diagrama copo de nieve que se hizo partiendo de los datos origen para estructurar el *Data Warehouse*.

A continuación comentamos las distintas tablas:

Usuario: Guardamos los usuarios por un *nickname* y un email.

Categoría: Formada por un *id* y el nombre de la categoría. Hay 4 tipos de categorías: Ocio, Relax, Business y Cultural.

Restaurante: *Id*, nombre y el coste. Añadiendo el precio del restaurante los usuarios que estén interesados en ese itinerario podrán saber una aproximación del gasto.

Alojamiento: Se seguirá la misma dinámica que en el restaurante.

Lugar: Guardaremos el nombre del lugar, longitud y latitud. El nombre es la identidad, ha de ser único.

Ciudad: Un *id*, el nombre de la ciudad y el lugar, ya que con este último hacemos referencia a la dimensión lugar.

Divisa: Los tipos de monedas que se utilizan según el destino que se visite.

Tráfico Web: Con esta dimensión contemplamos la fecha y el número de visitas que tiene cada itinerario.

Itinerario: Los itinerarios de la aplicación con su fecha de creación, una descripción, un coste total, la comisión que se lleva la empresa y los *id* de las dimensiones nombradas anteriormente.

Booking: En esta tabla guardamos la información que necesitamos sobre las reservas con sus correspondientes fechas.

Razón cancelación: Almacenamos el *id* y la razón por la que se canceló una reserva

Estado: El estado en el que se encuentra la reserva en ese momento.

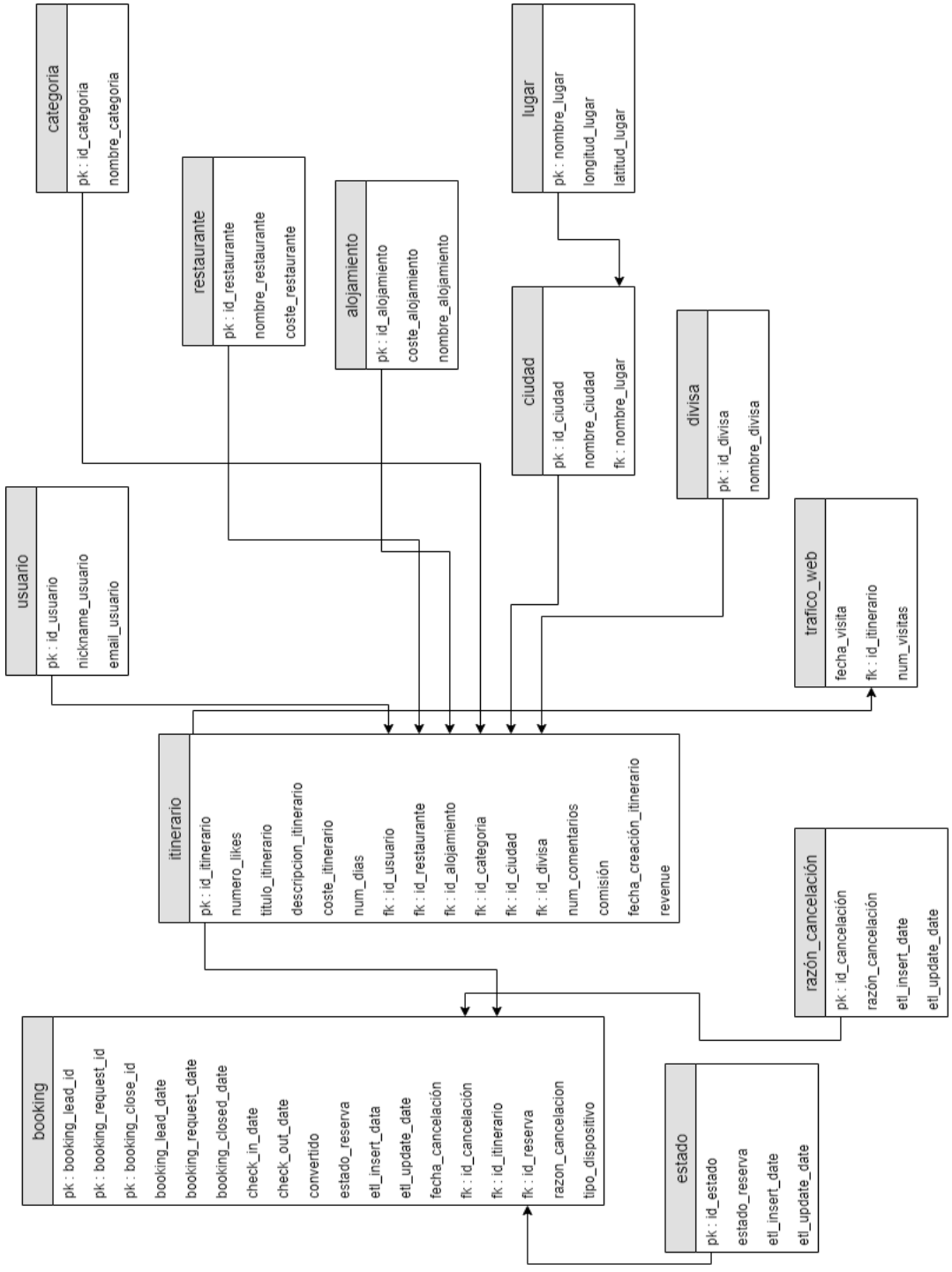


Figura 6.2: Modelo copo de nieve

6.3 Sprint 2: Creación del modelo de datos

En el *sprint* anterior hemos analizado los datos y los hemos estructurado para poder empezar a realizar la ETL, pero antes tenemos que crear las tablas en nuestra base de datos destino con sus correspondientes dimensiones y hechos. En la imagen 6.3 mostramos todas las tablas creadas en nuestra base de datos en SQLServer.

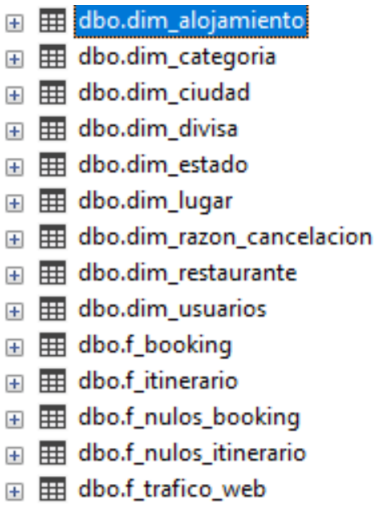


Figura 6.3: Tablas de la base de datos

Las dimensiones se han cargado en nuestro destino con datos del origen, menos el *id* que se auto-genera con `identity`; con este atributo se generan valores auto-incrementales que se inician en 1 y se incrementan automáticamente de 1 en 1. Esto se utiliza generalmente en códigos de identificación ya que crea valores únicos. Con lo cual de esta manera tendremos nuestra clave primaria.

Podemos observar que tenemos *ETL insert date* y *ETL update date* en todas nuestras tablas, estas son las fechas de creación y de actualización. En el primero guardaremos la fecha de inserción inicial. La de actualización coge el mismo valor hasta que ese campo se actualice.

```
create table dim_usuarios(
[id_usuario] int identity(1,1) primary key,
[nikname_usuario] varchar(20),
[email_usuario] varchar(50) not null,
[etl_insert_date] varchar(max),
[etl_update_date] varchar(max)
)
```

Listing 6.1: Ejemplo de creación de una tabla de dimensión

Por otro lado tenemos la tabla de itinerario, al igual que en las dimensiones auto-generamos la clave primaria y conectamos todas las dimensiones con ella mediante claves foráneas. En itinerario solamente tenemos la identificación, que al estar unida con la tabla de dimensiones se puede realizar un *join* accediendo al resto de los datos.

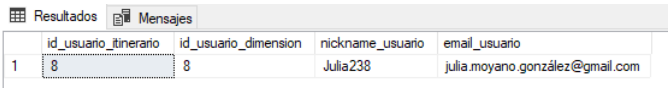
```
create table f_itinerario(
[id_itinerario] int identity(1,1) primary key,
[numero_likes_itinerario] int null,
[titulo_itinerario] varchar (50) not null,
[descripcion_itinerario] varchar (max) null,
[coste_itinerario] float null,
[num_dias] int null,
[id_usuario] int not null,
[id_restaurante] int not null,
[id_alojamiento] int not null,
[id_categoria] int not null,
[id_ciudad] int not null,
[id_divisa] int not null,
[num_comentarios] varchar (100),
[etl_insert_date] varchar(max),
[etl_update_date] varchar(max),
[fecha_creacion_itinerario] date,
Constraint FK_usuario Foreign key (id_usuario) references
    dim_usuarios(id_usuario),
Constraint FK_categoria Foreign key (id_categoria) references
    dim_categoria(id_categoria),
Constraint FK_alojamiento Foreign key (id_alojamiento) references
    dim_alojamiento(id_alojamiento),
Constraint FK_restaurante Foreign key (id_restaurante) references
    dim_restaurante(id_restaurante),
Constraint FK_ciudad Foreign key (id_ciudad) references
    dim_ciudad(id_ciudad),
Constraint FK_division Foreign key (id_divisa) references
    dim_divisa(id_divisa)
)
```

Listing 6.2: Tabla de hechos de itinerario

Por ejemplo, en la tabla de itinerario tenemos el *id usuario* número 8, haciendo *join* con la tabla de *dim usuarios* podremos saber el email y el *nickname* de ese usuario sin necesidad de tener todo cargado en la misma tabla. Esto será eficiente en las búsquedas. Con la siguiente consulta se pueden sacar los resultados del *id* número 8.

```
select a.id_usuario as id_usuario_itinerario, b.id_usuario as
    id_usuario_dimension, b.nickname_usuario,b.email_usuario
 from f_itinerario a inner join dim_usuarios b
 on a.id_usuario=b.id_usuario
 where a.id_usuario=8;
```

Listing 6.3: Ejemplo de un join entre una tabla de hechos y una de dimensiones


Resultados Mensajes

	id_usuario_itinerario	id_usuario_dimension	nickname_usuario	email_usuario
1	8	8	Julia238	julia.moyano.gonzález@gmail.com

Figura 6.4: Tablas de la base de datos

La tabla de *booking* se ha creado independiente de las dimensiones relacionadas con itinerario, en ella cargamos todos los datos respecto a las reservas. Como se muestra en el siguiente código se hacen dos claves foráneas a dos dimensiones relacionadas con la tabla de *booking*: *estado y razón cancelación.* En la primera de ellas mostramos los estados en los que se encuentra una reserva, en la otra, los motivos por los cual se puede cancelar.

```
create table f_booking(
[booking_lead_id] int not null,
[booking_request_id] int not null,
[booking_closed_id] int,
[booking_lead_date] date null,
[booking_request_date] date null,
[booking_closed_date] date null,
[check_in_date]  date null,
[check_out_date] date null,
[razon_cancelacion] varchar (100),
[tipo_dispositivo] varchar (20),
[estado_reserva] varchar (30),
[id_itinerario] int not null ,
[etl_insert_date] varchar(max),
[etl_update_date] varchar(max),
[id_cancelacion] int,
[id_reserva] int,
[fecha_cancelacion] date,
[convertido] varchar(5),
primary key(booking_lead_id,booking_request_id, booking_closed_id),
Constraint FK_itinerario Foreign key (id_itinerario) references
    f_itinerario(id_itinerario),
Constraint FK_cancelacion Foreign key (id_cancelacion) references
    dim_razon_cancelacion(id_cancelacion),
Constraint FK_estado Foreign key (id_reserva) references
    dim_estado(id_estado)
)
```

Listing 6.4: Creación de una tabla de hechos de booking

Independientemente de las tablas anteriores, que están conectadas entre si, hemos creado en nuestra base de datos dos tablas de auditoría que nos permiten controlar la información errónea: *nulos itinerario* y *nulos booking*. En ellas guardaremos los datos que no son correctos, aquellos que al realizar la transformación de los datos no pase algún filtro, tanto si está duplicado, como si es incorrecto.

Estas bases de datos tienen exactamente los mismos campos que las originales de itinerario y *booking*, ya que los campos que se guardarían serían exactamente los mismos pero con un valor incorrecto como un nulo. La creación de estas tablas nos daría la oportunidad de realizar una auditoría en el futuro.

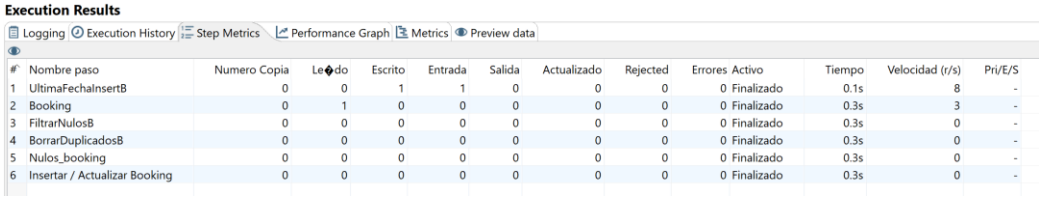
```
create table f_nulos_booking(
[booking_lead_id] int,
[booking_request_id] int,
[booking_closed_id] int,
[booking_lead_date] date,
[booking_request_date] date ,
[booking_closed_date] date ,
[check_in_date]  date ,
[check_out_date] date ,
[razon_cancelacion] varchar (100),
[tipo_dispositivo] varchar (20),
[estado_reserva] varchar (30),
[id_itinerario] int   ,
[etl_insert_date] varchar(max),
[etl_update_date] varchar(max),
[id_cancelacion] int,
[id_reserva] int,
[fecha_cancelacion] date,
[convertido] varchar(5))
```

Listing 6.5: Ejemplo de creación de una tabla de valores nulos

6.4 Sprint 3: Desarrollo del Data Warehouse

Empezaremos el desarrollo en Pentaho, se hace una transformación y los filtros necesarios para cargar posteriormente en SQLServer. El desarrollo de este *Data Warehouse* se basa en una carga incremental; esta carga es mucho más rápida que una carga completa pero tiene el inconveniente del mantenimiento. La carga incremental se realiza con una comparación entre la última fecha de inserción y el tiempo actual. Por ejemplo, si nuestra última inserción en la base de datos es del 16 de junio, y, la del registro que queremos cargar es anterior a esa fecha ,no nos permitirá cargarlo en la base de datos. En la imagen 6.5 se muestra que el paso

6 *insertar/actualizar* no ha insertado ningún registro en la base de datos.



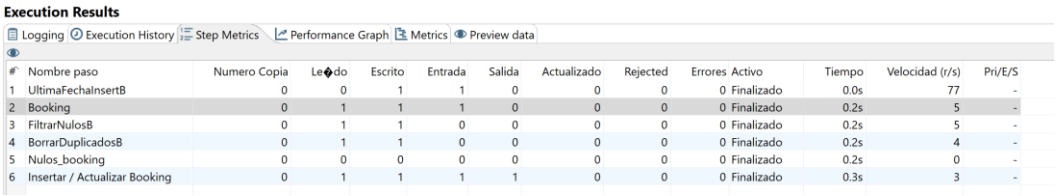
Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

#	Nombre paso	Numero Copia	Le◆do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
1	UltimaFechaInsertB	0	0	1	1	0	0	0	0	Finalizado	0.1s	8	-
2	Booking	0	1	0	0	0	0	0	0	Finalizado	0.3s	3	-
3	FiltrarNulosB	0	0	0	0	0	0	0	0	Finalizado	0.3s	0	-
4	BorrarDuplicadosB	0	0	0	0	0	0	0	0	Finalizado	0.3s	0	-
5	Nulos_booking	0	0	0	0	0	0	0	0	Finalizado	0.3s	0	-
6	Insertar / Actualizar Booking	0	0	0	0	0	0	0	0	Finalizado	0.3s	0	-

Figura 6.5: Ejemplo de carga incremental sin inserción en Pentaho

Por el contrarío si la fecha del registro que queremos cargar es superior al 16 de junio, se realiza la carga en la base de datos sin inconveniente. En la imagen 6.6 ,en el paso de *insertar /actualizar booking* se ha cargado el nuevo registro o la actualización de algún campo en el destino.



Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

#	Nombre paso	Numero Copia	Le◆do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
1	UltimaFechaInsertB	0	0	1	1	0	0	0	0	Finalizado	0.0s	77	-
2	Booking	0	1	1	1	0	0	0	0	Finalizado	0.2s	5	-
3	FiltrarNulosB	0	1	1	0	0	0	0	0	Finalizado	0.2s	5	-
4	BorrarDuplicadosB	0	1	1	0	0	0	0	0	Finalizado	0.2s	4	-
5	Nulos_booking	0	0	0	0	0	0	0	0	Finalizado	0.2s	0	-
6	Insertar / Actualizar Booking	0	1	1	1	1	0	0	0	Finalizado	0.3s	3	-

Figura 6.6: Ejemplo de carga incremental con inserción en Pentaho

Es necesario hacer una conexión con cada una de las bases de datos que queremos conectarnos, en nuestro caso con PostgreSQL y SQLServer. Estas conexiones se realizan una vez y se pueden utilizar las veces que sean necesarias.

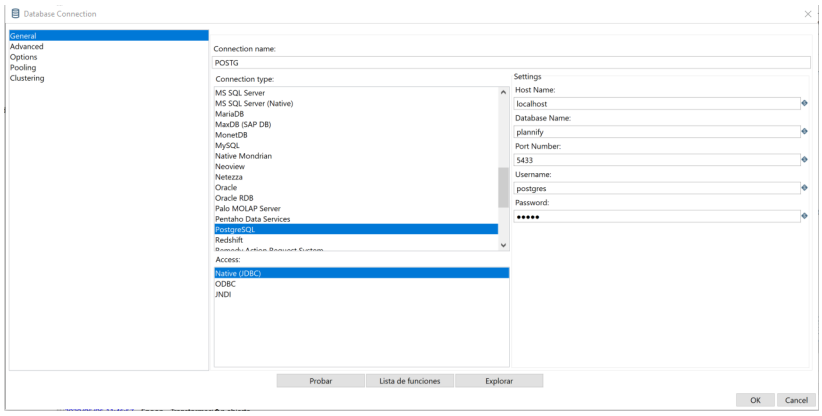


Figura 6.7: Conexión de Pentaho con PostgreSQL

A continuación vamos a explicar los componentes utilizados en una dimensión y las transformaciones necesarias para realizar la carga. El objetivo de las dimensiones es cargar los dis-

tintos campos que tengamos de cada uno para evitar guardar la información de forma redundante. En este caso guardaremos los distintos usuarios que tenemos, para evitar así guardar varias veces el mismo, ya que pudo crear más de un itinerario o reserva.

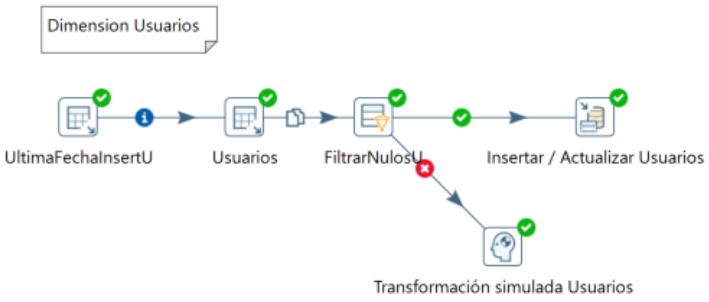


Figura 6.8: ETL de usuarios

En el primer componente del flujo anterior, *UltimaFechaInsertU* , aplicamos lo que definíamos anteriormente como carga incremental. Se coge la última fecha de inserción *(etl insert date)* que tenemos en SQLServer. Este dato lo obtenemos para hacer comprobaciones a la hora de insertar en el siguiente componente del flujo.

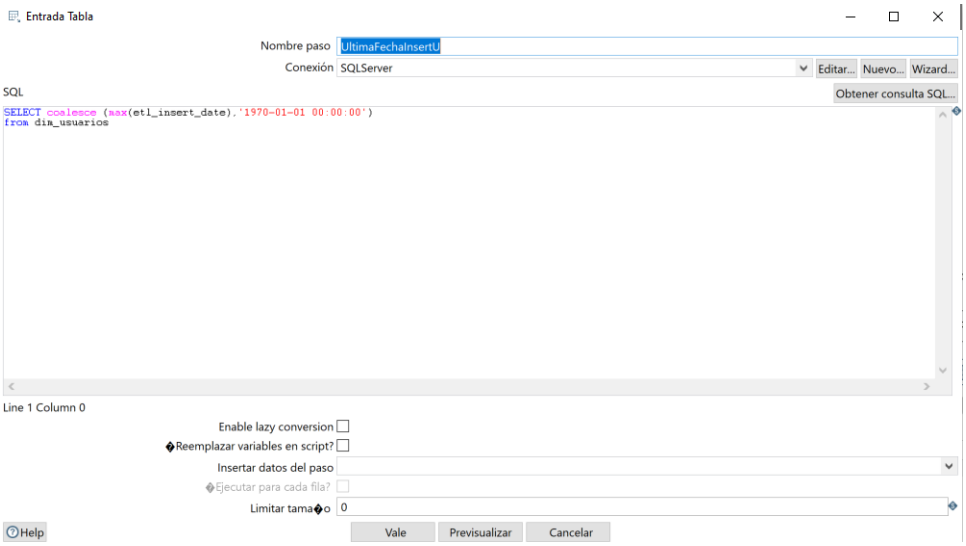


Figura 6.9: Selección de la fecha máxima

En el segundo componente llamado *Usuarios* se obtienen los datos del origen. Se extraen todos los campos que tenga la base de datos origen excepto:(1) *Id* que como hemos dicho anteriormente se genera automáticamente . (2) *etl insert* y *etl update* que se cargan con `current timestamp`; una función que devuelve la fecha de la base de datos actual.

En la primera inserción esta fecha será la misma, pero si se actualiza ese campo solo

se modificará la de actualización. Para recoger estos datos hacemos una comprobación y es que *sync data* (fecha origen) sea mayor que *etl insert date*, sino no se extraen los datos. Esto quiere decir que solo se insertan aquellos campos que tengan una fecha posterior a la máxima insertada en la base de datos destino.

Por ejemplo, en nuestro caso tenemos insertado en SQLServer la fecha máxima de inserción: 18/05/2020 y en PostgreSQL el 20/05/2020, de esta manera cogeríamos los datos del origen, sin embargo si la fecha de inserción fuera 22/05/2020 sería superior y no cogeríamos esos datos.

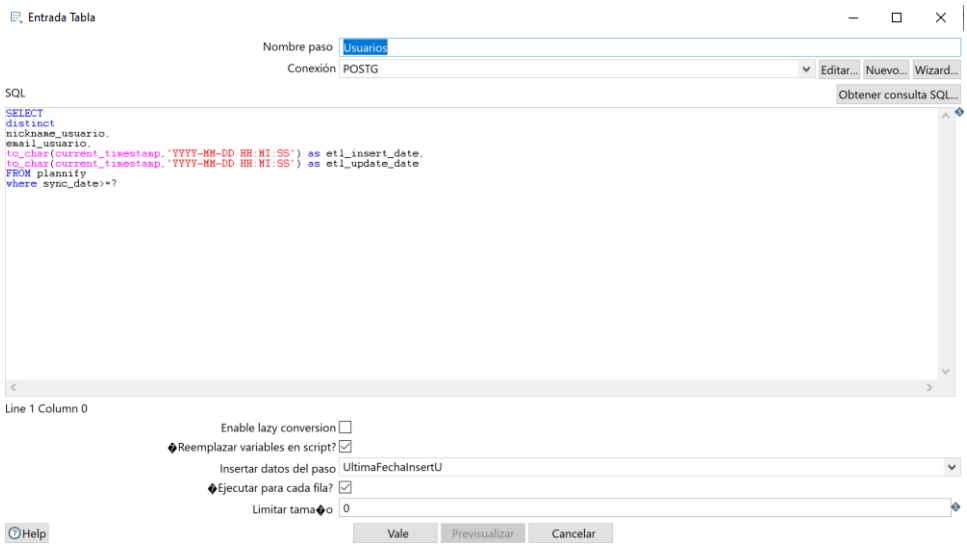


Figura 6.10: Entrada PostgreSQL

En el tercer componente, llamado *Filtrar Nulos U*, se utiliza un componente que nos ofrece Pentaho que se llama Filtrar Filas, con el se filtran los campos que no queremos que se inserten nulos. Este componente nos permite hacer distintas comprobaciones; si un campo es nulo, si no es nulo, si son iguales, distintos, si contiene algo, si son verdaderos, entre otras opciones. En función de esas comprobaciones , si ese campo es nulo, lo enviamos a un componente que se llama Transformación Simulada ,que no hace nada, o lo pasamos al siguiente componente Insertar/Actualizar que explicamos en el siguiente paso.

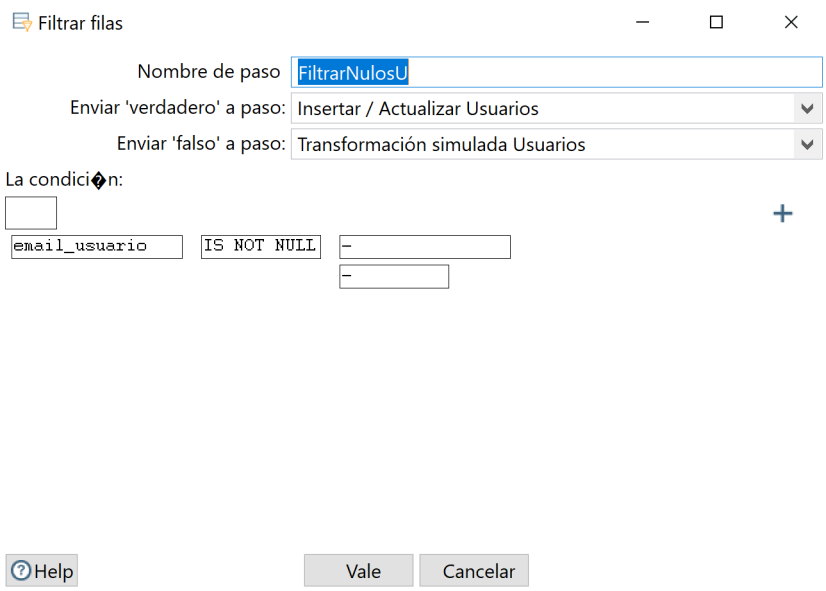


Figura 6.11: Comprobar nulos

El último componente es el que se llama *Insertar/Actualizar*, es un componente que tiene esta herramienta que nos permite, partiendo de la tabla que ya tenemos almacenada en nuestro destino, utilizar unos valores de búsqueda en función de si se cumple insertar u actualizar los campos. En el caso de usuario , comprobamos si el email que hemos recogido es igual a alguno ya insertado en la base de datos origen, si es así actualizamos los campos que queramos. En este caso actualizamos los campos cuya información ha sido refrescada en las fuentes orígenes. Utilizando insertar/actualizar, no es necesario volver a cargar todos los datos cada vez que queramos hacer una modificación, solo los campos necesarios.

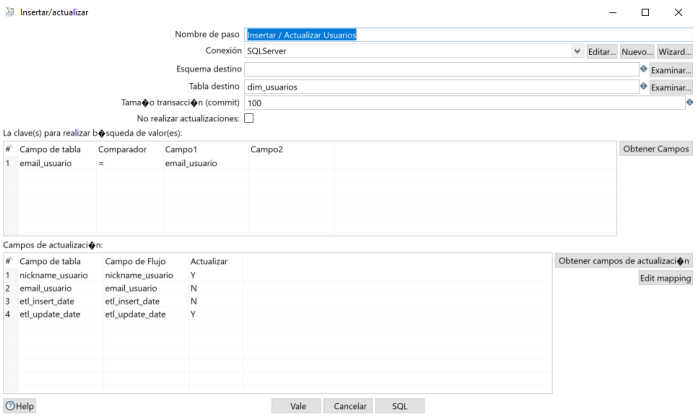


Figura 6.12: Insertar o actualizar usuarios

La extracción, transformación y carga de todas las dimensiones se han realizado de la misma manera.

6.5 Sprint 4: Desarrollo del Data Warehouse - Hechos

En este sprint continuamos utilizando la carga incremental comentada en el capítulo anterior, pero se realizan algunos cambios respecto a las tablas de dimensiones. En las tablas de hechos se controlan los datos duplicados, borrándolos para evitar redundancia en nuestra base de datos destino.

Por otro lado, en vez de perder la salida si encontramos alguna anomalía, la guardamos en una tabla de base de datos. De todos los itinerarios que se han creado, algunos serán descartados, porque no cumplen todos los requisitos para ser cargados en la base de datos, pero guardarlos nos sirve para analizar otro tipo de información y mejorar la calidad del dato con la que se aprovisionan las tablas a nivel empresarial en un futuro.

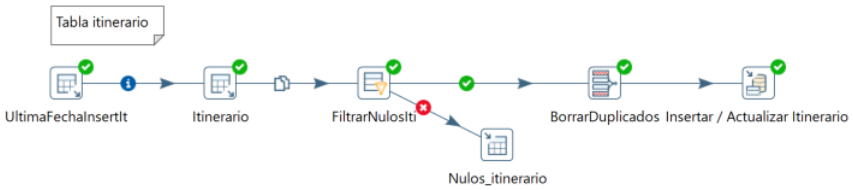


Figura 6.13: Flujo de itinerario

En los primeros pasos, al igual que en los flujos de carga de las dimensiones, se obtiene la última fecha ,se recogen los datos del origen y se filtran los nulos aplicando en este caso el uso de tablas de auditoría, para almacenar aquellos registros que no han llegado completos, como es el caso de valores nulos en campos que no pueden serlo. Es decir, si no existen nulos se pasa al siguiente paso,*Borrar duplicados*, por el contrario se almacenan en la tabla *nulos booking*. En detalle, filtramos todos los *id*, ya que no se puede cargar un *id* nulo. En el caso de que se cumpla alguna de las condiciones se guarda en la tabla de nulos itinerario. Se añade a nuestro flujo de *booking*, *Filas únicas*, este componente de Pentaho nos permite borrar aquellas filas que se encuentren duplicadas.

Como se muestra en la imagen 6.14, se eliminan aquellos que aparezcan duplicados al nivel de los campos que se utilizan. De esta manera se evita que nos salgan dos filas exactamente iguales. La carga de filas duplicadas proporciona datos incorrectos para el siguiente paso que se va a realizar.

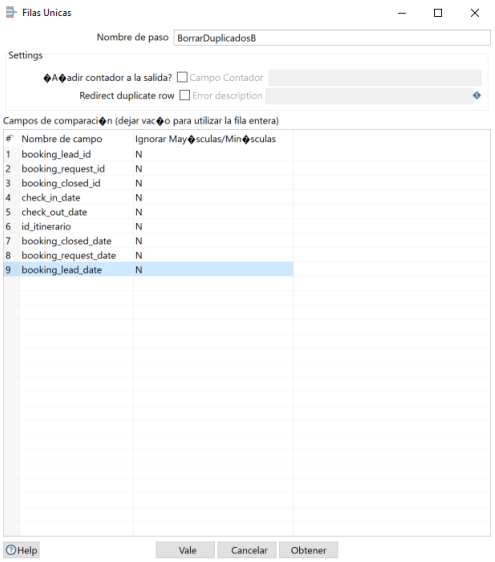


Figura 6.14: Componente filas únicas de Pentaho

Para insertar en la tabla de nulos se utiliza el componente llamado *Salida de Tabla*. Con este componente se guarda en la base de datos todos los datos que tengamos incorrectos porque hayan sido nulos, pero sin realizar una carga incremental. Esto sirve para detectar aquellos campos que traen nulos e intentar reparar los procesos que los están dejando pasar. Esto puede ser útil para controlar datos anómalos que entran desde la aplicación, y ayudar a los desarrolladores a encontrar fallos que pueden no estar detectando.

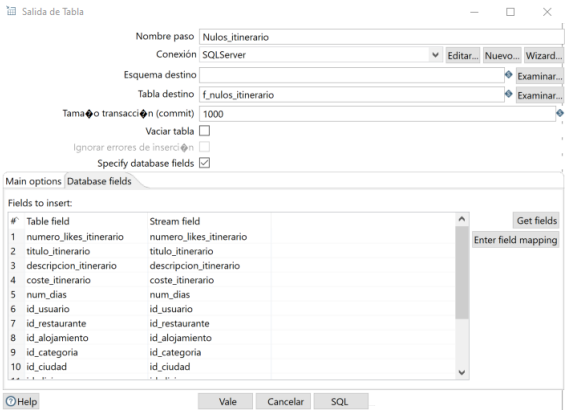


Figura 6.15: Componente Salida Tabla para insertar en la tabla de nulos de SQLServer

El flujo total de todas las dimensiones y hechos se presenta como muestra la figura 6.16.

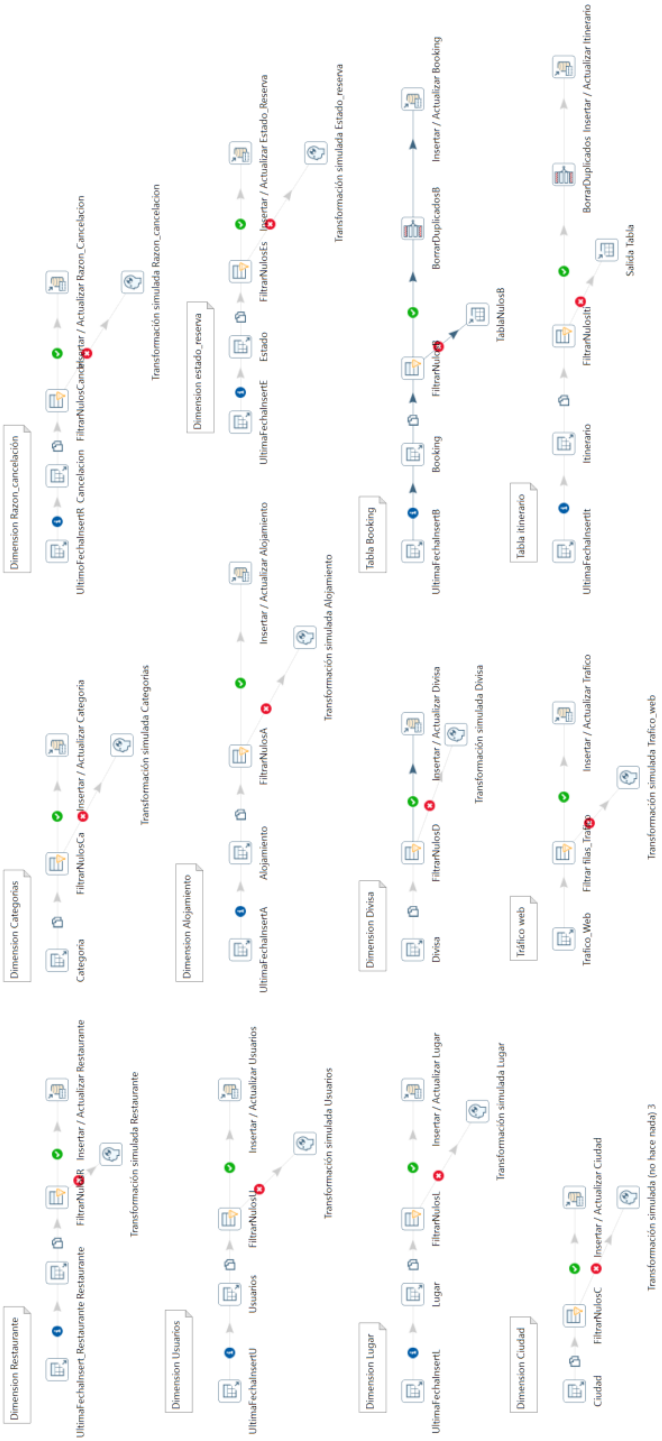


Figura 6.16: Representación de todos los flujos

6.5.1 Pruebas

Tras realizar la carga de los datos, se realizan pruebas para ver si los datos se han insertado correctamente y si los filtros se han aplicado. Comprobamos si los resultados que están en PostgreSQL son iguales a los que llegan finalmente a cargarse en SQLServer.

En la imagen 6.17 se puede observar todas las dimensiones y hechos que tenemos y las comprobaciones que se han hecho en cada una de ellas.

	Pentaho	SQLServer	
dim_restaurante	15	15	
dim_usuarios	101	101	
dim_lugar	15	15	
dim_ciudad	15	15	
dim_categoria	4	4	
dim_alojamiento	15	15	
dim_divisa	5	5	
f_trafico_web	101	100	1nulo
dim_razon_cancelacion	5	5	
dim_estado	5	5	
f_itinerario	101	100	1nulo
f_booking	101	100	1nulo

Figura 6.17: Pruebas realizadas

A continuación se muestran dos figuras (6.18 y 6.19), una es el resultado de Pentaho y otra lo que se ha cargado en SQLServer tras realizar los flujos. En la primera de ellas podemos observar que de los 101 registros solamente se añaden 15. Estos son los 15 restaurantes distintos que se añaden a la dimensión asegurándonos que no entra ningún dato anómalo. En la segunda imagen vemos que se han cargados correctamente en SQLServer.

Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

#	Nombre paso	Numero Copia	Le◆do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
1	UltimaFechaInsert_Restaurante	0	0	1	1	0	0	0	0	Finalizado	0.0s	50	-
2	Restaurante	0	1	101	101	0	0	0	0	Finalizado	0.2s	513	-
3	FiltrarNulosR	0	101	101	0	0	0	0	0	Finalizado	0.2s	495	-
4	Insertar / Actualizar Restaurante	0	101	101	101	15	86	0	0	Finalizado	0.3s	336	-
5	Transformación simulada Restaurante	0	0	0	0	0	0	0	0	Finalizado	0.2s	0	-

Figura 6.18: Resultado del flujo de restaurante

Resultados | Mensajes

	id_restaurante	nombre_restaurante	coste_restaurante	etl_insert_date	etl_update_date
1	1	El colmado	265,020138037405	2020-06-07 12:04:12	2020-06-07 12:04:12
2	2	El Buda	90,51606388773	2020-06-07 12:04:12	2020-06-07 12:04:12
3	3	La barbacoa de Berlin	327,293319397027	2020-06-07 12:04:12	2020-06-07 12:04:12
4	4	Taberna el Sur	413,719666417054	2020-06-07 12:04:12	2020-06-07 12:04:12
5	5	Wellies	125,167665560978	2020-06-07 12:04:12	2020-06-07 12:04:12
6	6	SushiMou	276,333863464642	2020-06-07 12:04:12	2020-06-07 12:04:12
7	7	Cebo	412,377049154396	2020-06-07 12:04:12	2020-06-07 12:04:12
8	8	El jardin	87,1978300252124	2020-06-07 12:04:12	2020-06-07 12:04:12
9	9	LaLanda	372,356552914718	2020-06-07 12:04:12	2020-06-07 12:04:12
10	10	Mero	467,593743916119	2020-06-07 12:04:12	2020-06-07 12:04:12
11	11	Harold´s Fried Chicken	290,773715349878	2020-06-07 12:04:12	2020-06-07 12:04:12
12	12	Tony Roma´s	355,306534387219	2020-06-07 12:04:12	2020-06-07 12:04:12
13	13	Cherry	393,309480823047	2020-06-07 12:04:12	2020-06-07 12:04:12
14	14	El capricho	141,049130867042	2020-06-07 12:04:12	2020-06-07 12:04:12
15	15	Punto MX	134,577887382231	2020-06-07 12:04:12	2020-06-07 12:04:12

Figura 6.19: Datos de la dimensión restaurante en SQLServer

En el caso de la tabla de hechos de *booking* , se han realizado las mismas pruebas, con la diferencia de que se comprueba que los valores que encuentre nulos se guardan en una tabla de nulos de SQLServer. En el componente *Nulos booking* nos dice que un valor es nulo y la carga total a la tabla es 100. Si alguno de los datos no coincide o desaparece se tendría que revisar el flujo para saber donde se perdió exactamente el dato.

Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

#	Nombre paso	Numero Copia	Le◆do	Escrito	Entrada	Salida	Actualizado	Rejected	Errores	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
1	UltimaFechaInsertB	0	0	1	1	0	0	0	0	Finalizado	0.0s	333	-
2	Booking	0	1	101	101	0	0	0	0	Finalizado	0.2s	577	-
3	FiltrarNulosB	0	101	101	0	0	0	0	0	Finalizado	0.2s	567	-
4	BorrarDuplicadosB	0	100	100	0	0	0	0	0	Finalizado	0.2s	543	-
5	Nulos_booking	0	1	1	0	1	0	0	0	Finalizado	0.2s	5	-
6	Insertar / Actualizar Booking	0	100	100	100	100	0	0	0	Finalizado	0.2s	415	-

Figura 6.20: Resultado del flujo de Booking

6.6 Sprint 5: Visualización - Overview

En esta fase se comienza la parte de visualización con Power BI, a mayores de las tablas que se cargan de SQLServer, creamos *Calendar* y *Medidas*, donde se añaden las métricas que se fueron creando.

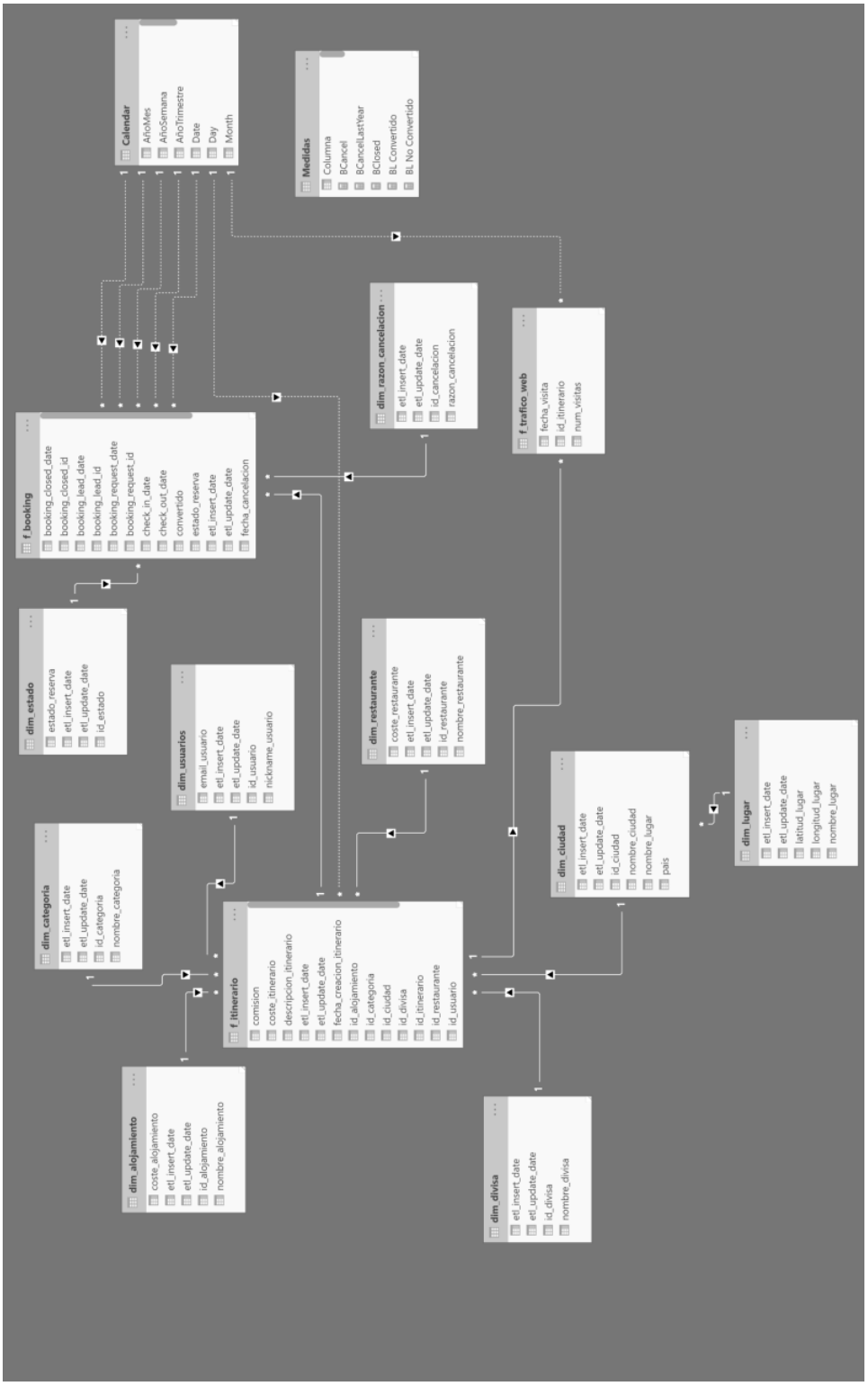


Figura 6.21: Diagrama Power BI

En la imagen 6.21 se muestran dos tipos de relaciones, por un lado tenemos las activas que son relaciones permanentes que forman el modelo de datos,funcionan de manera que cada tabla esta unida por un *inner join*. Por lo que cualquier métrica calculada sobre esta tabla se verá afectada por esta relación.

En la figura 6.22 se muestra la tabla de *booking* con una dimensión estado con una dirección activa por *id* estado.

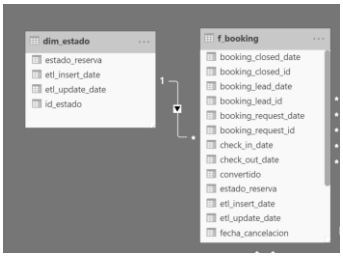


Figura 6.22: Relación activa

Por otro lado tenemos las inactivas, aquellas que vemos en el gráfico con líneas despuntadas. Estas se activan con el uso de una función *dax*, `userelationship`, que se debe utilizar dentro del desarrollo de cada *métrica* del cuadro de mando.

```
1 USERELATIONSHIP('Calendar'[Date],f_booking[fecha_cancelacion])
```

Listing 6.6: Ejemplo de la función userelationship

En el caso de nuestras métricas, no todas deben ser calculadas con las mismas fechas, por eso se utiliza este tipo con *calendar*, como se muestra en la imagen 6.23.

Se pueden ver distintas relaciones inactivas dependiendo de las distintas fechas, el calendario esta relacionado con la fecha de *booking lead*, *booking request*, *booking closed* y *fecha de cancelación*.

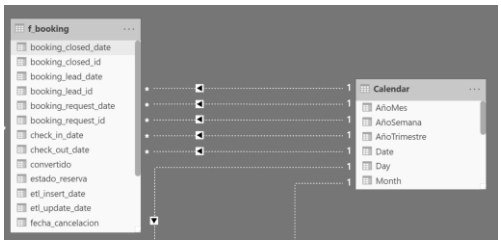


Figura 6.23: Relación inactiva

Para representar la información solicitada de forma visual se crean tres pantallas, la primera de ellas, que se muestra a continuación, es una visión general en la que se pueden ver diferentes métricas.

Figura 6.24: Overview

En cada una de las pestañas que se han creado se utilizan filtros, para que el usuario pueda seleccionar sus búsquedas en función de lo que le interesa. A nivel empresarial puede ser interesante solamente buscar unos años en concreto, el histórico global de los cuatro años o simplemente un mes en particular. Para estos filtros se selecciona lo que es llamado en Power BI *Segmentación de datos* . Respecto a figura 6.24 podemos observar los distintos tipos de filtros que se aplicaron, los menús desplegables (rojo), las listas (azul) o aquellos que pueden realizar búsquedas en un rango (naranja).

En el primer *gráfico de columnas agrupadas* ,se controla el tráfico web ,pudiendo ver la información agrupada en diferentes periodos de tiempo. De esta manera se puede saber cuando la aplicación recibe más visitas. Se puede interactuar con los gráficos utilizando la función `resumir` y `drill-down` (siguiente valor de la jerarquía) que nos permite navegar a través de na jerarquia de fechas, conformada por diferentes periodos de tiempo (año, trimestre, mes, semana y día)

En la figura 6.25 se observan las flechas que le permiten al usuario ir navegando por ella. Con esto se tiene la ventaja de que el usuario puede ver los diagramas sin necesidad de crear uno distinto para trimestre, año, mes, semana o día.

Figura 6.25: Número de visitas

Falling rate es el ratio de caída, que es lo que se muestra en este *gráfico de líneas*. Un *booking request* puede llegar a *booking close* o no, esto son los principios básicos para calcular el *falling rate*. Esto es interesante para el usuario porque puede analizar el porcentaje de reservas que se han cancelado y evaluar la situación. Si el ratio de reservas canceladas es mayor en el año actual que el anterior, se puede considerar que algún cambio que se ha realizado para mostrar a nuestros clientes no va según lo esperado. Es un gráfico `YoY` *(Year Over Year)*, se compara la misma métrica respecto al año anterior, para calcularla se utiliza `sameperiodlastyear`.

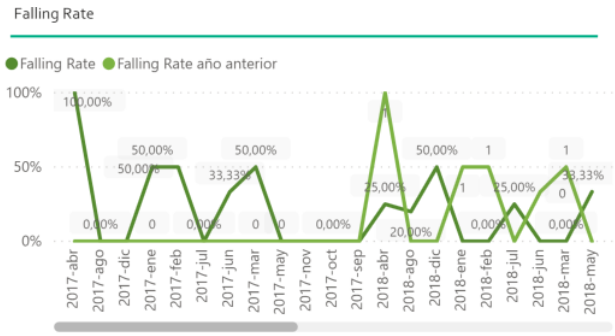


Figura 6.26: Gráfica en la que se calcula el Falling Rate

Con el siguiente KPI 6.27 tenemos las opciones de ir recorriendo la jerarquía de fechas o ciudades en el eje X. De esta manera, el usuario si lo desea puede llegar a escoger ciudades en el eje X, y, conocer cuantos itinerarios se han creado en esa ciudad, por el contrario, escoger fechas en la jerarquía y conocer el número de itinerario según determinadas fechas.

Con este gráfico podemos analizar rápidamente las ciudades en las que se han creado más itinerarios, e intentar ofrecer una mejora de *marketing* en otras sin olvidarnos de seguir creciendo en las mejor posicionadas.

Por ejemplo,en Barcelona se han creado 5 itinerarios y en Londres 8. Sin embargo en Casablanca solamente se han creado dos. Podemos potenciar esos itinerarios intentando captar mas clientes que se interesen en ellos.

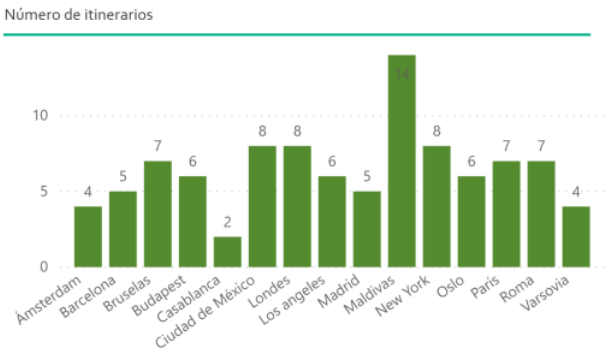


Figura 6.27: Número de itinerarios

A continuación haciendo referencia a *Número de viajes* 6.28 , tenemos un *gráfico de líneas* en el que se muestra cuantos viajes se han realizado en cada fecha en comparación con el año anterior. Así podemos conocer si nuestras reservas se han incrementado o decrementado respecto al año anterior.

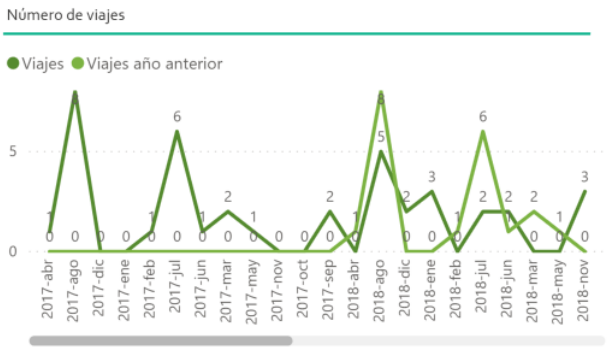


Figura 6.28: Número de viajes

Con el siguiente código se observa una métrica mediante la cual se utiliza la función `sameperiodlastyear` , de la que ya hemos hablado anteriormente, y `userelationship`, que nos sirve para crear la unión entre las fechas. La última función activa esta relación inactiva para crear la conexión entre *calendar* y *booking*.

```
ViajesLastYear = CALCULATE(
COUNT(f_booking[booking_request_id]),
USERELATIONSHIP('Calendar'[Date],f_booking[check_in_date]),
SAMEPERIODLASTYEAR('Calendar'[Date])
)
```

Listing 6.7: Métrica para el calculo del número de viajes

6.7 Sprint 6: Booking Lead y Booking Request

6.7.1 Booking Lead

La siguiente pestaña creada, es sobre los *booking lead*. En ella se muestra una visión general al comienzo de las reservas donde se puede analizar si las reservas llegan a realizarse, o las categorías en las que más se prioriza según la época del año.

Figura 6.29: Booking Lead

Se puede mostrar interés en algunos itinerarios, pero no llevar a cabo una reserva, en este punto es donde contemplamos cuantos se han convertido y cuantos no.

En la tabla de hechos de *booking* tenemos un campo, llamado convertido, que nos indica si un *booking lead* ha llegado a ser *booking request*. También se utiliza `userelationship` para conectar la fecha de *booking lead* con *calendar* , ya que se trata de una relación inactiva.

```
BL No Convertido=
CALCULATE(
DISTINCTCOUNT(f_booking[booking_lead_id]),
(f_booking[convertido="No"]),
USERELATIONSHIP('Calendar'[Date],f_booking[booking_lead_date)
)
```

Listing 6.8: Métrica de un booking lead no convertido

Con dos métricas, una la del código mostrado anteriormente, y otra con el campo convertido a *"si"* ,podemos crear el *gráfico por columnas agrupadas* ,en el que se muestra según las fechas seleccionadas cuantos *booking lead* han llegado a convertirse y cuantos no. El usuario puede realizar `resumir` y `drill-down` navegando por la jerarquía y así conocer la tendencia que hay en el ratio de conversión entre el *booking lead* y el *booking request*.

Por ejemplo, en la siguiente imagen podemos observar que en enero del 2017 la empresa tiene mas *booking lead* que se han convertido de los que no, pero, sin embargo, en marzo del 2019 es al contrario. Según el mes se puede realizar un *marketing* más llamativo que otro, a petición de los usuarios de la aplicación.

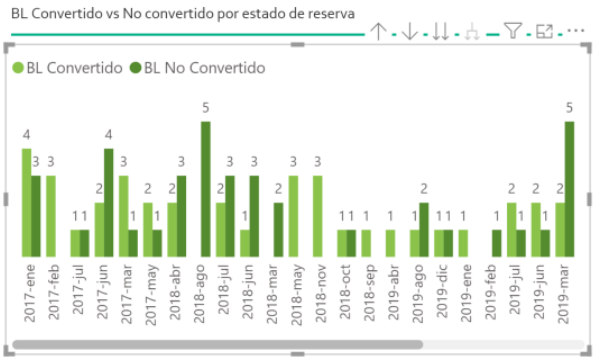


Figura 6.30: BL convertido vs BL no convertido

Los viajes se pueden clasificar en distintas categorías dependiendo de la época del año, unos viajes serán más recurrentes que otros.

Es frecuente que en épocas como verano los usuarios reserven más viajes de relax u ocio, mientras que el resto del año el nivel de viajes por negocios o culturales aumenten. La empresa podrá tomar decisiones de marketing en función de la categoría, ya que los viajes dependiendo de la estación pueden variar. Podemos observar por meses cuantos viajes de cada categoría se realizan para que la empresa

pueda comercializarlos en función de las fechas.Lo mostramos con un *gráfico de columnas apiladas* 6.31
.

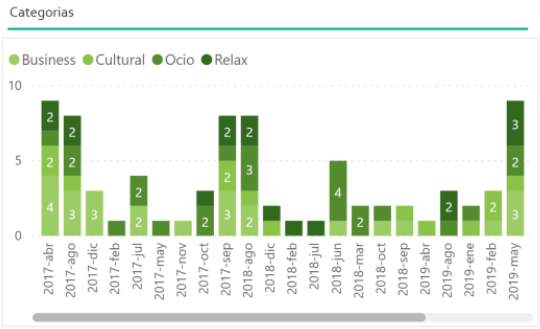


Figura 6.31: Categorías

Para finalizar este informe, se muestra en esta pestaña una tabla que nos dá información general de las reservas. En ella mostramos: la fecha de *booking lead*, el identificador, el estado en el que se encuentra, en qué fechas se realiza y un *nickname* de usuario. Con la tabla se puede analizar la antelación con la que se reservan viajes porque conocemos el momento inicial en el que un itinerario llamó la atención, de esta manera podemos mostrar los más interesantes. Estos estudios se pueden realizar gracias a que existen datos de otros años y se ha ido creando un histórico. Por otro lado podemos conocer los estados de reserva que más destacan en ese momento y si algo resulta anómalo.


Booking Lead

BL date	BL id	estado_reserva	Fecha de entrada	Fecha de salida	nickname
lunes, 2 de enero de 2017	248	Cancelado	domingo, 5 de febrero de 2017	miércoles, 8 de febrero de 2017	Alejandro721
domingo, 1 de enero de 2017	381	Pendiente	miércoles, 15 de marzo de 2017	jueves, 23 de marzo de 2017	Sonia451
lunes, 2 de enero de 2017	209	Aprobado	miércoles, 15 de marzo de 2017	sábado, 25 de marzo de 2017	Juan Antonio147
jueves, 2 de febrero de 2017	230	Pendiente	sábado, 15 de abril de 2017	miércoles, 19 de abril de 2017	María638
jueves, 23 de marzo de 2017	130	Cancelado	miércoles, 17 de mayo de 2017	viernes, 19 de mayo de 2017	José853
lunes, 2 de enero de 2017	315	Cancelado	martes, 6 de junio de 2017	viernes, 16 de junio de 2017	Mercedes283
jueves, 22 de junio de 2017	197	Pendiente	martes, 4 de julio de 2017	viernes, 14 de julio de 2017	María Isabel839
lunes, 2 de enero de 2017	347	Alternativa-cancelada	viernes, 14 de julio de 2017	lunes, 17 de julio de 2017	Mercedes283

Figura 6.32: Tabla que nos muestra información de los booking lead

6.7.2 Booking request

La siguiente y última pestaña creada, es dedicada exclusivamente a estudiar las *booking request*, analizar los distintos estados en los que se encuentra la reserva, el porcentaje que tenemos o las cancelaciones, y así ayudar a la toma de decisiones.

Figura 6.33: Booking Request

La figura 6.33 muestra 4 gráficos; el primero de ellos es un *gráfico de columnas apiladas* que muestra los estados de una reserva. Se puede conocer cuantas tenemos en cada estado, de esta manera, a nivel de empresa se puede saber cuantas han sido aprobadas, cuantas canceladas y cuales después de una alternativa se aceptan o se cancelan. Dependiendo del estado, el usuario puede decidir si ha de mejorar ya que existen más estados alternativos o cancelados que aprobados directamente. Con las alternativas lo que se pretende es medir la eficiencia de los agentes de *booking* a la hora de ofrecerlas.

Figura 6.34: Solicitudes por estados de reserva

En el gráfico de lineas 6.35 se pueden conocer el número de cancelaciones en cada periodo de tiempo y comparar si, un año respecto al anterior, ha fluctuado considerablemente.

La empresa puede adentrarse en profundidad y estudiar los motivos por los que fue cancelada esa reserva, por ejemplo, si es por motivos ajenos, o, por el contrario, tiene la posibilidad de realizar cambios respecto a ciertos itinerarios.

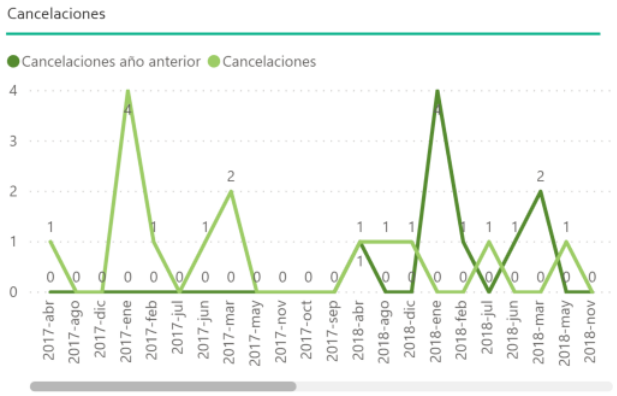


Figura 6.35: Cancelaciones

En el *gráfico circular* 6.36 mostramos 3 métricas distintas para conocer el porcentaje de las reservas. *Booking request* son aquellas reservas que están en proceso de ser aceptadas por nuestros agentes, es decir, todos los usuarios que han decidido realizar su viaje a través de la aplicación. Además, se puede saber cuantas de ellas se han cancelado teniendo en cuenta que algunas reservas han podido cerrarse y cancelarse después por algún motivo.

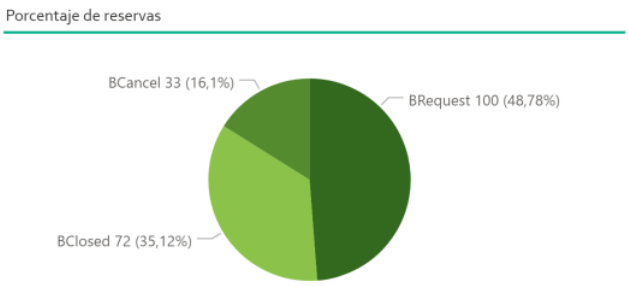


Figura 6.36: Porcentaje de Reservas

En la figura 6.37 observamos un gráfico llamado *gráfico anillo*, con el cual el usuario conoce cuantas de las reservas que han tramitado han sido cerradas, cuantas canceladas y a cuantas se les ha ofrecido una alternativa. En la métrica creada usamos el campo de estado reserva para saber el estado actual de cada reserva. Es una manera rápida de que el usuario, a grandes rasgos, tenga una visión general de como han ido sus reservas. Podemos ver que el número de reservas cerradas es elevado y las canceladas bastante pequeño. Sin embargo sigue quedando un gran número de reservas que se han cerrado después de ofrecer una alternativa, lo que significa que las propuestas iniciales no están siendo del todo acertadas y se pueden mejorar.

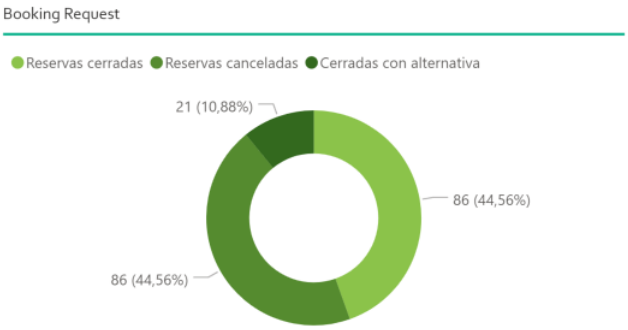


Figura 6.37: Booking Request

Pruebas

Se ha comprobado si los datos que llegan a Power BI son los mismos que se han cargado anteriormente en SQLServer tras realizar la ETL.

	Pentaho	SQLServer	PowerBI	
dim_restaurante	15	15	15	
dim_usuarios	101	101	101	
dim_lugar	15	15	15	
dim_ciudad	15	15	15	
dim_categoria	4	4	4	
dim_alojamiento	15	15	15	
dim_divisa	5	5	5	
f_trafico_web	100	100	100	1nulo
dim_razon_cancelacion	5	5	5	
dim_estado	5	5	5	
f_itinerario	100	100	100	1nulo
f_booking	100	100	100	1nulo

Figura 6.38: Pruebas realizadas Power BI

Para comprobarlo se realiza una consulta a la base de datos de cuantos campos se tiene de cada una y en Power BI se realiza una métrica para que nos permita saber que campos son en total.

Como anteriormente se ha probado con una dimensión y una tabla de hechos, se van a realizar las mismas comprobaciones.

La dimensión restaurante: en la herramienta nos muestra los mismos campos que los guardados en la dimensión de SQLServer

Figura 6.39: Número de restaurantes en Power BI

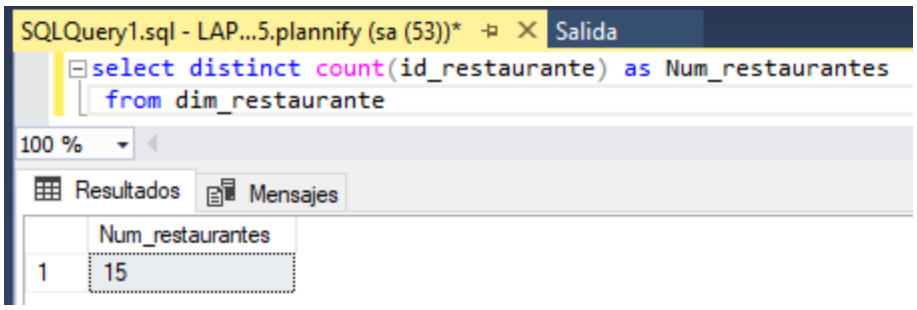


Figura 6.40: Número restaurantes SQL

Tabla de *booking*: Corroboramos que los mismos registros que se han cargado en SQLServer tras las transformaciones han llegado a Power BI sin ninguna pérdida.

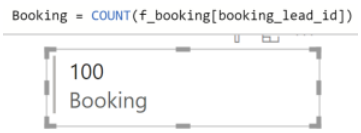


Figura 6.41: Número booking Power BI

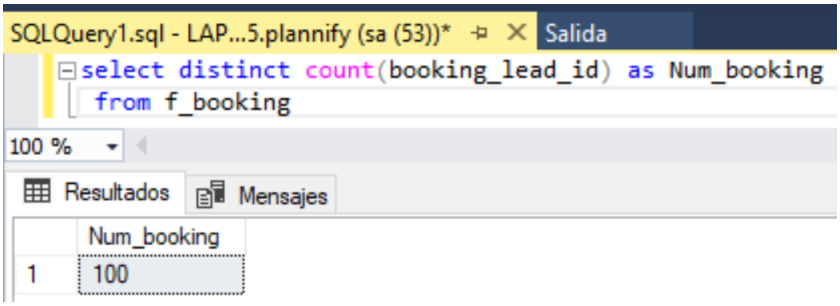


Figura 6.42: Número de booking en SQL

6.8 Publicar informes

Una vez se realizan los análisis y se crean todos los informes, se procede a subirlos a un repositorio virtual gestionado por Microsoft. Esto nos proporciona una gran ventaja, ya que se puede crear distintos usuarios que puedan acceder a nuestro cuadro de mando sin necesidad de tenerlo en local y con el beneficio de poder acceder desde cualquier lugar. De esta manera siempre pueden acceder a la información actualizada.

Power BI tiene una licencia de desarrollo gratuita, Power BI Desktop, pero ofrece una versión Power BI Pro. Que provee al usuario la facilidad de crear un espacio llamado, **área de trabajo**, donde se pueden subir los cuadros de mando que en este caso serán empleados para este proyecto, porque, a diferencia de la versión de escritorio, nos permite editar y crear informes en colaboración con otros miembros del equipo con la misma licencia, y también, añadir capa de seguridad a los informes para administrar de forma controlada los accesos.

Para la publicación de informes es necesario crear una cuenta de Microsoft 365 E5. Esta herramienta nos permite gestionar usuarios y roles para que puedan acceder a la información de diferentes maneras. En el caso del administrador, puede gestionar los distintos usuarios y el control de los cuadros de mando, por el contrario el usuario tiene acceso para analizar y navegar a través de los informes. En este caso se han creado usuarios con diferentes roles

Por otro lado en `app.powebi.com` se publican los informes que queramos para que los usuarios naveguen por ellos. El usuario inicia sesión y podrá acceder al cuadro de mando que se muestra en la siguiente imagen 6.43 de las 3 pestañas en este caso. La persona que este navegando a través de estos informes tiene la posibilidad de exportar estos informes a PDF, a Power Point o imprimirlos, así como, realizar los filtros para conocer la información que le sea útil en el momento actual.

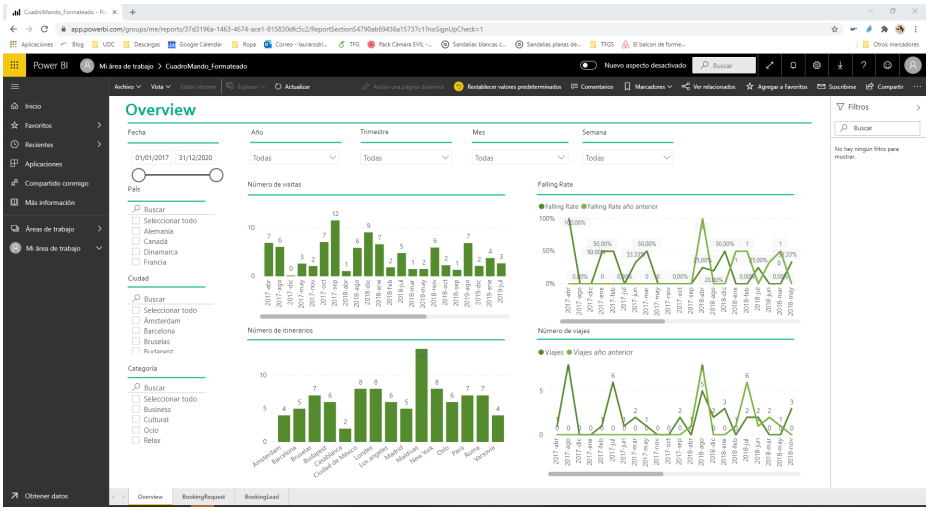


Figura 6.43: Informe publicado en powerbi.com

La publicación se realiza desde la aplicación de escritorio seleccionando un **área de trabajo** en la cual queremos que esté el informe. El administrador puede realizar la actualización de los informes de forma manual desde la aplicación de escritorio de Power BI, sustituyendo el publicado anteriormente por el nuevo con los datos actualizados. De esta manera puede actualizarlo cuando se precise. Por otro lado tiene la opción de realizar una actualización programada, se puede programar con una frecuencia y una franja horaria sin que el administrador este pendiente de las publicaciones.

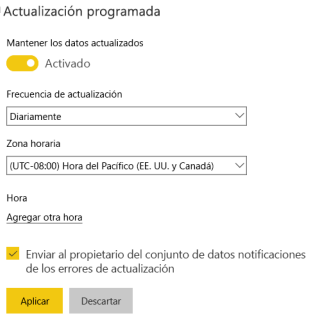


Figura 6.44: Ejemplo de actualización programada

Capítulo 7

Conclusiones

En este proyecto se ha llevado a cabo la aplicación de *Business Intelligence* en el ámbito de viajes demostrando la potencia de estos sistemas.

El objetivo fundamental ha sido la creación de un *Data Warehouse* para conseguir una visualización de información que ayudase a mejorar el rendimiento empresarial en el sector de los viajes. Se proporciona al usuario el conocimiento necesario mediante gráficos sobre los datos de los itinerarios y reservas. Para ello se han identificado los factores más importantes que afectan al rendimiento mediante el estudio de distintas técnicas de inteligencia de negocio. Gracias al sistema que se ha desarrollado y a los análisis llevados a cabo, se ha podido determinar los dos factores primordiales.

El primero de ellos es el número de itinerarios, ya que la oferta de ellos es clave a la hora de ofrecerlo a distintos usuarios, para cubrir un abanico más amplio de posibilidades. Los itinerarios más visitados pueden variar en función de los años, siempre habrá destinos que permanezcan con mas afluencia en ciertos meses, y otros que mejoren por el *márketing* ofrecido.

En segundo lugar, las reservas son clave para conocer el estado de nuestro negocio, ya que nos permite saber el momento en el que se encuentran. El estado de una reserva se puede ver afectado en función de distintos factores, es importante que se haya analizado este campo porque es determinante para la toma de decisiones.

7.1 Objetivos alcanzados

Al iniciar el proyecto se han propuesto unos objetivos,estos se han cumplido aplicando las técnicas de la inteligencia de negocio en el sector de los viajes.

En primer lugar, se ha hecho el tratamiento de datos mediante un flujo ETL, para que en la base de datos destino no entren datos anómalos que interfieran después en nuestro cuadro de mando.

En segundo lugar, los informes se han creado con los gráficos esperados en función de las métricas utilizadas, de esta manera el usuario puede hacer un análisis de los datos para conseguir la información que desea, y maximizar la rentabilidad. Cabe destacar que para la creación de nuevos gráficos son necesarias nuevas tablas con más información, ya que con la actual hay ciertas métricas que no se pueden calcular, por ejemplo, impacto en los ingresos esperados en el año 2020, en comparación con los percibidos por la situación actual.

7.2 Líneas futuras

Al inicio del proyecto se barajaron distintas ideas para el desarrollo del *Data Warehouse* pero que finalmente no pudieron ser incluidas, pero que sería interesante considerar si se continuase el trabajo:

1. Mantenimiento del cuadro de mando actual. Es necesario un mantenimiento sobre el cuadro de mando y los procesos que lo aprovisionan,ya que en el caso de error imprevisto, sería necesario aplicar acciones correctivas para asegurar la veracidad de los datos visualizados.
2. Realizar un estudio de aquellos elementos anómalos que se han guardado en las tablas de nulos. A partir de estos datos, se realizará un análisis para determinar si esos datos se han perdido durante el desarrollo de la ETL.
3. En base a las quejas de los usuarios en los distintos itinerarios, analizar los puntos más débiles e intentar mejorarlos.
4. Crear nuevos informes con campos que todavía no se han tratado, pero de los que mantenemos registros,como es el caso de la tabla divisa, que actualmente no está siendo explotada.

7.3 Valoración personal

Personalmente, la realización de este proyecto me ha supuesto un aprendizaje continuo. Principalmente porque me ha dado la oportunidad de adquirir conocimiento y habilidad sobre herramientas dentro del mundo de *business intelligence* que no conocía, y a la vez aplicar los conocimientos adquiridos a lo largo de la titulación.

BI es una parte de la informática que está en expansión, cada día se utiliza más para conocer el estado de las empresas, conocer sus necesidades de información y darle soluciones útiles que necesitan.

Para concluir, resalto la satisfacción que he obtenido tras obtener conocimiento en nuevas tecnologías utilizadas en este proyecto que me sirven para una nueva etapa profesional.

Lista de acrónimos

KPI *Key Performance Indicator.*

DAX *Data Analysis Expressions.*

OLAP *On-Line Analytical Processing*

OLTP *Online Transaction Processing*

ETL *Extract Transform Load*

BI *Business Intelligence*

IoT *Internet of Things*

DDL *Data definition language*

DML *Data manipulation language*

Glosario

BI Procesos, tecnologías y estrategias enfocadas a ayudar en la toma de decisiones basándose en la gestión del conocimiento.

Métrica Una medida cuantificable utilizada para evaluar el éxito de una empresa.

Backend Parte ”oculta” donde se realiza la configuración global de los diferentes procesos ETL

Frontend Capa ”visible” donde se realizan los informes

DataMart Es una base de datos especializada en el almacenamiento de información de un área

Data warehouse Base de datos o repositorio destinado a almacenar información de distintas fuentes. La base sobre la que se nutren los sistemas de análisis orientados a facilitar la toma de decisiones.

Internet of things Interconexión digital de objetos cotidianos con internet. Esta relacionado con Big Data por la cantidad de información que se genera a través de dispositivos.

Bibliografía

[1] J. L. Cano, "Business intelligence: Competir con información." [En línea]. Disponible en: http://itemsweb.esade.edu/biblioteca/archivo/Business_Intelligence_competir_con_informacion.pdf

[2] C. Howson, *Successful Business Intelligence: Unlock the Value of BI and Big Data*, 2nd ed. Mc Graw Hill Education.

[3] R. Wrembel, "Data warehouse and olap." [En línea]. Disponible en: https://books.google.es/books?hl=es&lr=&id=8HSSrj-3dRYC&oi=fnd&pg=PR1&dq=Data+Warehouses+and+OLAP+Concepts,+Architectures+and+Solutions&ots=fgO1-lZEgC&sig=QY5PKoKg4foOqHf8hdBFa6PGSb8#v=onepage&q=Data%20Warehouses%20and%20OLAP%20Concepts%2C%20Architectures%20and%20Solutions&f=false

[4] A. Maydanchik, *Data Quality Assessment.* Technics publications, LLC.

[5] "Modelo estrella." [En línea]. Disponible en: https://docs.microsoft.com/es-es/power-bi/guidance/star-schema

[6] "Kpi." [En línea]. Disponible en: https://docs.microsoft.com/es-es/analysis-services/multidimensional-models/key-performance-indicators-kpis-in-multidimensional-models?view=asallproducts-allversions

[7] "Magic quadrant for analytics and business intelligence platforms." [En línea]. Disponible en: https://info.microsoft.com/ww-landing-2020-gartner-magic-quadrant-for-analytics-and-business-intelligence.html?LCID=EN-US

[8] "Qlik." [En línea]. Disponible en: https://www.qlik.com/us/

[9] "Tableau." [En línea]. Disponible en: https://www.tableau.com/es-es

[10] "Foro pentaho." [En línea]. Disponible en: https://help.pentaho.com/Documentation/8.2/Products/Data_Integration

[11] "Power bi." [En línea]. Disponible en: https://powerbi.microsoft.com/es-es/

[12] "Scrum." [En línea]. Disponible en: https://www.scrum.org/resources/blog/que-es-scrum

[13] "Trello." [En línea]. Disponible en: https://trello.com/laurarodriguez388/boards